SIEMENS

BS2000
DMS Disk Processing

Reference Manual

Edition November 1985 (BS2000 V8.0A)

Order No. U805-J-Z55-5-7600
Printed in the Federal Republic of Germany
3580 AG 3860.8 (4470)

Order No.:
U805-J-Z55-5-7600

A ring binder for this manual may be obtained at a cost of DM 4,20 stating the order number U1225-J-Z18-1

SIEMENS

BS2000
OMS Disk Processing

Reference Manual

Edition November 1985 (BS2000 V8.0A)

Order No. U805-J-Z55-5-7600
Printed in the Federal Republic of Germany
2E/0786-4E/0 (4.20)

Order No.:
U805-J-Z55-5-7600

A ring binder for this manual may be obtained at a cost of DM 4.20, stating the order number U1224-J-Z18-1.

BS2000

DMS Disk Processing

Reference Manual

Edition November 1985 (BS2000 V8.0A)

Order No. U805-J-Z55-5-7600
Printed in the Federal Republic of Germany
3580 AG 3860.8 (4470)

PREFACE

This manual describes the BS2000 Data Management System provided for the processing of files on magnetic disk. The corresponding description for tape files can be found in the "DMS Tape Processing" manual.

The manual is intended for the BS2000 user who is looking for information concerning file processing. The descriptions of the macro interface and access methods are required by the Assembler programmer.

- Chapters 1 and 2 contain an overview of the DMS functions.
- Chapter 3 contains a description of the DMS commands.
- Chapter 4 contains a description of the DMS macros.
- Chapter 5 explains the access to files. This chapter also includes a description of the service macros.
- A description of the UPAM, ISAM, SAM and EAM access methods, complete with their action macros, is provided in chapter 6.
- The appendix contains information on:
  - conventions
  - DMS DSECTs
  - DMS error codes and error listing
  - shareable private disk
  - definition of common expressions.

The amendments and additions that have been made since the previous edition of this manual are summarized in the List of Amendments 1, following this preface.

Other manuals are referred to in the text by their abbreviated titles. The full title of each publication mentioned is given at the end of the present manual under the heading "Literature". This list also includes the "Druckschriftenverzeichnis" (list of Siemens publications) and the "BS2000 System Reference Guide".

To help us continue to improve our publications, please send us your comments, requests and suggestions using the pink reply forms provided.

Editorial Department K D ST PM 2

U805-J-Z55-5-7600

PREFACE

This manual describes the BS2000 Data Management System provided for the processing of files on magnetic disk. The corresponding description for tape files can be found in the "DMS Tape Processing" manual.

The manual is intended for the BS2000 user who is looking for information concerning file processing. The descriptions of the macro interface and access methods are required by the Assembler programmer.

- Chapters 1 and 2 contain an overview of the DMS functions.
- Chapter 3 contains a description of the DMS commands.
- Chapter 4 contains a description of the DMS macros.
- Chapter 5 explains the access to files. This chapter also includes a description of the service macros.
- A description of the UPAM, ISAM, SAM and EAM access methods, complete with their section macros, is provided in chapter 6.
- The appendix contains information on:
  - conventions
  - DMS OSECTs
  - DMS error codes and error listing
  - shareable private disk
  - definition of common expressions.

The amendments and additions that have been made since the previous edition of this manual are summarized in the list of Amendments 1, following this preface.

Other manuals are referred to in the text by their abbreviated titles. The full title of each publication mentioned is given at the end of the present manual under the heading "Literature". This list also includes the "Druckschriftenverzeichnis" (list of Siemens publications) and the "BS2000 System Reference Guide".

To help us continue to improve our publications, please send us your comments, requests and suggestions using the pink reply forms provided.

Editorial Department K D ST PM 2

LIST OF AMENDMENTS 1

The following amendments have been made since publication of the
Revision September 1984 (BS2000 V7.6A)
and incorporated in the
Edition November 1985 (BS2000 V8.0A):

Functions (chapter 2):

Topic	Amendment
Temporary files	Temporary files cease to exist as soon as the job is terminated (LOGOFF). However, they are specified in commands and used just like normal DMS files. Temporary files are identified by a prefix in the name.
MPVS (Multiple Public Volume Set)	Multiple public volume sets consist of a number of independent public volumes (pubsets) on a single system. All the disks in a pubset are recognized by the system as a single unit. Pubsets are distinguished by their catalog identifiers (catid).

DMS commands (chapter 3):

Command	Amendment
CATALOG	It is possible to convert a temporary file to a permanent file and vice versa.
COPY	By means of the new operand WRITE it is possible to specify whether or not an existing file is to be overwritten.
ERASE	It is possible to erase temporary files and up to 99 SYSLST files.
FILE	Specification of a temporary file possible. New device types: D3480 D348E
FSTATUS	The new operand FROM enables display from a catalog with the specified "catid". Symbol specification is possible for the operands "catid" and "filename". In the case of temporary files, the internal file name is always specified.
IMPORT	The REPLACE operand has been extended by the specification ABS: Importing takes place if the catalog entries have the same VSN as the disk to be imported. The new operand PVSID defines the catalog which is to accommodate the catalog entries.
SEC-RES SECURE	As a result of the new device management, the old SECURE command has been revised and the new SECURE-RESOURCE-ALLOCATION command has been introduced.
SHOW-DEVICE-STATUS	This new command provides information on device occupancy and volume monitoring.
SHOW-DISK-DEFAULTS	This new command requests the default values for disk parameters.
SHOW-DISK-STATUS	This new command provides information on disk occupancy.
SHOW-MOUNT-PARAMETER	This new command requests information on mount parameters.
SHOW-RESOURCE-ALLOCATION	This new command provides information on resource allocation and ongoing operator actions.

DMS macros (chapters 4 and 5):

Macro	Amendment
CATAL	Cf. the CATALOG command.
COPY	The new operand WRITE determines whether or not an existing file is to be overwritten.
DSTATUS	This new macro displays information on the peripheral devices.
FSTAT	This macro has been brought into line with the FSTATUS command; the old FSTAT format is retained.
IMPORT	Cf. the IMPORT command.
RDTFT	The description of the operands has been revised.
FCB	A new operand enables the specification of options lists: GLODEF second attempt to read; WARN warning return codes are stored without execution being interrupted.

Error codes (Appendix A.3):

The error codes were listed by means of the IDEMS macro.

BHS macros (chapters 6 and 8):

Macro	Amendment
CATAL	Cf. the CATALOG command.
COPY	The new operand WRITE determines whether or not an existing file is to be overwritten.
DSTATUS	This new macro displays information on the peripheral devices.
FSTAT	This macro has been brought into line with the FSTATUS command; the old FSTAT format is retained.
IMPORT	Cf. the IMPORT command.
RDTFT	The description of the operands has been revised.
FCB	A new operand enables the specification of optional files. CLOSEF second attempt to read; WARN warning return codes are stored without execution being interrupted.

Error codes (Appendix A.3):

The error codes were listed by means of the IDEMS macro.

CONTENTS

1 INTRODUCTION

1.1 POSITION OF THE DMS IN BS2000

The Data Management System (DMS) is a self-contained component within the Control System, alongside the Executive, the teleprocessing system and the system services. It serves both the named components of the Control System and the users of BS2000.

The diagram below shows the position of the Data Management System within the overall system.

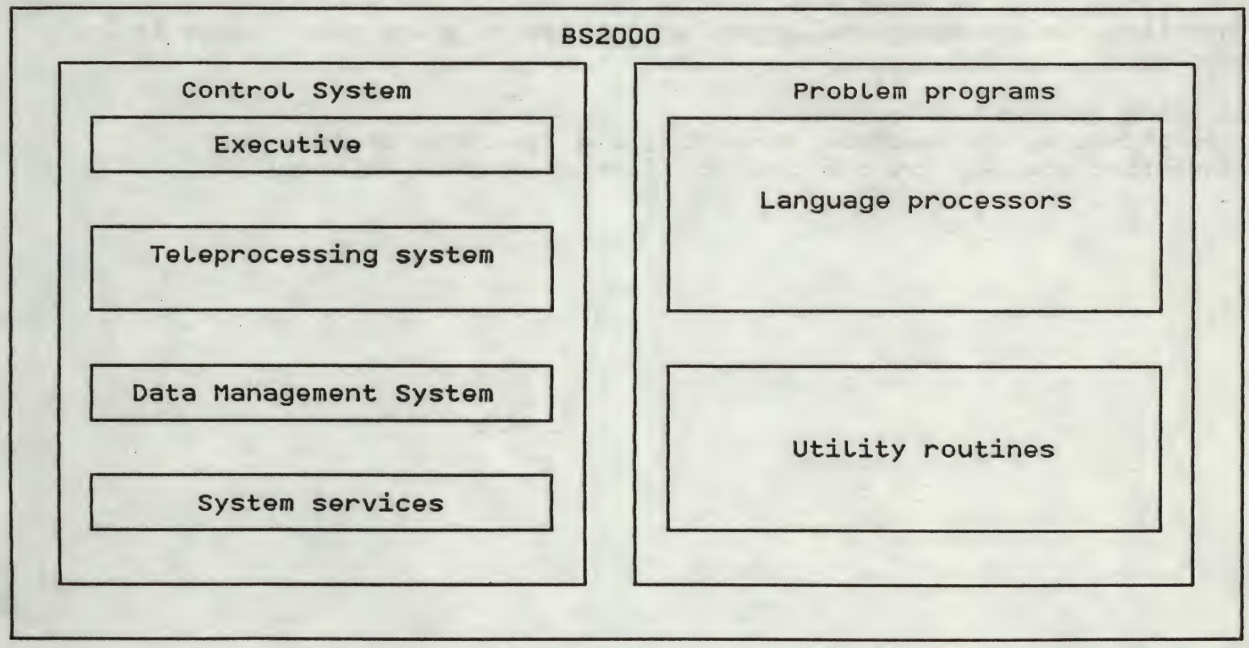


Fig. 1-1: Position of the Data Management System within the overall system

1.2 OVERVIEW OF THE FUNCTIONS OF THE DMS

The Data Management System offers the user "information containers", called files, for storing and retrieving data, as well as a variety of functions necessary for working with files:

- Maintaining files:
  This includes, for example, creating, erasing and copying files, and defining protection attributes.

- Assigning files to programs:
  This includes the option of delaying specification of the files for a program until immediately before execution of the program.

- Accessing files:
  This includes opening and closing files, and reading from and writing to files.

- Protecting files against unauthorized access:
  This includes checking the access authorization every time a file is opened.

- Managing devices and volumes:
  This includes, for example, reserving and releasing devices and requesting volumes (in the case of files on private volumes).

1.3 SUMMARIES OF DMS COMMANDS AND MACROS

The Data Management System provides the user with a command interface and a macro interface.

Almost all of the functions of the command interface are also available in the macro interface.

However, the macro interface contains additional functions which the command interface does not provide (e.g. access to files).

The following sections provide summarized information on the functional scope of the command and macro interfaces.

1.3.1 Commands and Macros for Maintaining Files

Command	Macro	Summary of function
CATALOG	CATAL	creates or updates a catalog entry.
COPY	COPY	copies a file or a file generation group.
ERASE	ERASE	erases one or more of a user's files.
FILE	FILE	defines all attributes of a file and creates the catalog entry and an entry in the Task File Table (TFT) for this file; reserves storage space.
FSTATUS		transfers information from a file's catalog entry to the system file SYSOUT.
	FSTAT	transfers catalog entries in part or in whole to a user area (without passwords).
IMPORT	IMPORT	creates catalog entries for existing files on private disks.
VERIFY	VERIF	restores corrupted files.

1.3.2 Commands and Macros for Assigning Files to Programs within a Job

Command	Macro	Summary of function
CHANGE	CHNGE	changes the file link name of a file (in the Task File Table, TFT).
FILE	FILE	assigns a file link name to a file and creates an entry for this name in the Task File Table (TFT).
RDTFT	RDTFT	indicates which file link names are assigned to which files in the TFT.

1.3.3 Commands and Macros for Managing Private Devices

Command	Macro	Summary of function
RDTFT	RDTFT	supplies information concerning devices and volumes associated with a certain file.
RELEASE	REL	releases devices.
SECURE [-RESOURCE -ALLOCATION]		requests resources (files, volumes, devices).

1.3.4 Macros for Accessing Disk Files

Preparing access:

Macro	Summary of function
FCB	creates a File Control Block. This control block serves to - inform the system which file is to be processed. - convey file characteristics to the system (if they cannot be taken from an entry in the Task File Table or a catalog entry). - indicate to the program the current characteristics of the file after it has been opened.
FCBAD	generates the code in such a way that the FCBs may be outside the base register in the case of symbolic addressing.
OPEN	opens a file.

Terminating access:

Macro	Summary of function
CLOSE	closes a specified file or all files open at that time.

Handling errors and contingencies:

Macro	Summary of function
EXLST	transfers to the DMS addresses at which processing is to be continued in the event of errors or task contingencies.
EXRTN	returns control to the DMS after executing certain EXLST exits and informs the DMS how processing should be continued.

Accessing file contents:

UPAM

Macro	Summary of function
PAM	passes jobs to UPAM (e.g. lock block, read block, write block, etc.).

Summaries

The following macros are of significance only if UPAM is used in conjunction with eventing. Then the underlined macros are mandatory, the others optional, according to the problem definition.
(For a full description of the macros see the "Executive Macros" manual.)

ENAEI	allocates an event item to a user task.
DISEI	deallocates a user task from an event item.
CHKEI	checks the status of an event item.
SOLSIG	sends a request to an event item.
ENACO	enables a contingency definition, i.e. allows it to control contingency tasks.
DISCO	nullifies a contingency definition, i.e. prohibits it from controlling contingency tasks.
LEVCO	changes the priority level of a contingency task or of the basic task.
RETCO	terminates a contingency task.
CONTXT	provides read or write access to the register set and program counter (the "context") of an interrupted contingency task or of the basic task.

SAM

Macro	Summary of function
GET	retrieves the next record from a file.
PUT	writes a record to the end of a file.
PUTX	replaces a record previously retrieved by a GET macro and updated by the program.
RELSE	bypasses the remaining memory area of a buffer.
SETL	enables positioning to the beginning of the file or to the end of the file, or to the point determined by the retrieval address.

ISAM

Macro	Summary of function
ELIM	eliminates a record from the file.
GET	retrieves the next record in a file.
GETFL	searches within the specified range for the next record which satisfies the flag criteria, and retrieves it.
GETKY	retrieves a record containing the specified key.
GETR	retrieves the next record toward the beginning of the file.
INSRT	inserts a record in a file.
ISREQ	cancels a set ISAM lock.
OSTAT	informs the user about the number of jobs sharing the file and the type of shared access (read, write).
PUT	writes a record to the end of a file.
PUTX	replaces a record previously retrieved by a GET, GETFL, GETKY or GETR macro.
RETRY	resets the ISAM pointer, after the PGLOCK exit has been performed, to the correct position required by a following GET, GETR or GETFL macro, and optionally repeats the macro which could not be performed because of the locked data block.
SETL	enables positioning to the beginning of the file, to the end of the file, or to the point determined by the key.
STORE	stores a record in the location determined by the specified key. If multiple records with the same key are not permitted, the record to be stored overwrites any existing record having the same key.

EAM

Macro	Summary of function
EAM	passes jobs to EAM (e.g. create EAM file, read block, write block, close file).

1.3.5 Macros for Generating Symbolic Addresses for Control Blocks and Operand Lists

Macro	Summary of function
IDCAT	CATALOG (CATAL) operand list
IDCE	Catalog entry
IDCEG	Catalog entry, addition for file generation groups
IDCEX	Catalog entry extension
IDCHA	CHANGE (CHNGE) operand list
IDCOP	COPY macro operand list
IDECB	UPAM event control block
IDEE	Catalog entry, extent list
IDEMS	DMS error messages This macro generates a list of EQUs which describes the error messages of the DMS modules (cf. Appendix A.3, "DMS Error Codes")
IDERS	ERASE macro operand list
IDFCB	FCB (P1 section) Note: this DSECT describes all FCB formats.
IDFST	FSTATUS (FSTAT) macro operand list
IDIMP	IMPORT macro operand list
IDMCB	EAM control block
IDOST	File information about opened files of the user
IDPFL	FILE macro operand list
IDPFX	FILE macro operand extension
IDPPL	UPAM operand list
IDREL	RELEASE (REL) macro operand list
IDVT	Volume label entry
IDVRF	VERIF macro operand list
DMADR	Task File Table
DMARD	RDTFT macro operand list

2 FILES, VOLUMES AND DEVICES IN BS2000

2.1 FILES

2.1.1 General

Information that is to be processed in a computer must be collected in such a way as to produce logically coherent groups in machine-readable form.

The smallest logical unit is a **character** (letter, digit). A number of characters form an **expression** (name, street, tax bracket, bank statement).

Expressions which describe the same object (a specific customer, the required spare part, a particular colleague) are combined in a **record.**

Several records of the same type which are stored and managed as a single unit form a **file.**

Every file in BS2000 has a name that distinguishes it unequivocally from all other files, and typical dp-specific characteristics such as storage structure, access method, type of volume, protection attributes, etc.

The CPU and the peripheral devices do not know the logical structure of a file. For the sake of the I/O system and the device controllers files are divided into transfer units known as **blocks.**

A block can contain either exactly one or several logical records. However, a logical record can also be distributed over several physical blocks if the logical block comprises more than one physical block.

Data records consist of a (usually fixed) number of **bytes**, i.e. characters.

Examples of files:

- Conventional input/output data used by programs
- Source programs
- Module libraries
- Textual information to be created and processed by the file editor
- Command sequences (ENTER and DO procedures)
- Executable programs

Files

How are logical records stored in a file?

For storing logical records in and retrieving them from a file, the DMS access methods ISAM and SAM are used (for PAM, a record is the same as a block). It should be noted that a file can be processed only using the access method it was created with.

Exception:
A disk file can be read with the access method UPAM, regardless of which access method it was created with.

What are the different types of file?

- Permanent files

  Permanent files are cataloged files on external storage media, whose names and characteristics are kept by the DMS in the system catalog, and which are therefore known to the system. These files thus become resources that can be used by more than one job. However, only in special cases can a number of parallel jobs write to the same file simultaneously.

- EAM files

  EAM files are stored in the common area SYSEAM and are implicitly deleted when the job is terminated (LOGOFF).

- Temporary files

  In contrast to permanent files, a temporary file ceases to exist once the job that created it is terminated (LOGOFF). However, temporary files can be specified in commands and used just like normal DMS files (FCBTYPE, RECFORM, etc.).

  Temporary files are defined by a prefix in the file name. This prefix can be either of the special characters "#" or "@" and is defined at system generation time (class 2 option TEMPFILE). The system administrator can provide information on which character has been selected.

  Characteristics of temporary files:

  - Temporary files can be PAM, SAM or ISAM files.
  - Temporary files do not belong to the allocated public space, i.e. a user with no public space contingent can still create temporary files.
  - As the file names of temporary files are job-specific, there is no chance of collision between temporary files with the same name but belonging to two different jobs that are active at the same time.
  - In the event of errored system termination, the temporary files of the jobs active at the time remain in the catalog, as no LOGOFF can be performed. These files are deleted when the pubset is imported (IMCAT command).
  - The file names of temporary files must not exceed 31 characters, including the special character for the prefix.
  - Temporary files cannot be created on private disk.
  - Temporary file generation groups are not supported.

- Temporary files are always created on the user's standard pubset under the user's ID. It is neither possible to enter a specification under a foreign user ID or with a non-standard "catid", nor is it possible to access temporary files with RFA.

- Temporary files are kept in the file catalog under an internal file name: S.TMP.nnnn.filename (where "nnnn" is the TSN). This internal file name is always specified in information commands/macros (e.g. FSTATUS, RDTFT), in the ERASE command and in the system messages.

Note:

The use of internal file names and their incorporation into programs/procedures eliminates the possibility of portability with other operating systems.
Implementation via a real file with special characteristics is not a component of the user interface.
This can be modified at any time.

- Symbolic system files

  These are input and output files which are automatically assigned to every user job in BS2000.

  An example of a symbolic system file is SYSCMD, which is the data source from which the system takes commands for the associated job.

Which types of files are managed by the Data Management System?

The DMS manages only cataloged and temporary files, and only processing involving such files is described in the present manual. Operations using symbolic system files are described in the "Control System Command Language" and "Executive Macros" manuals.

Mapping of files onto volumes

The DMS maps files onto volumes. The type of mapping depends on:

- which volumes are used (tape or disk),

- how a file is to be accessed (e.g. sequential access or indexed sequential access).

The choice of volume therefore immediately limits the possible modes of access. Further details relating to this may be found in the chapter on access methods.

The DMS maintains appropriate tables so that a file mapped onto a volume may be reconstructed and its contents accessed at any time. For cataloged files, these tables are contained in the catalog entry. The tables for temporary files are located in a part of system memory.

2.1.2 File Nomenclature

Each cataloged file has a file name which distinguishes it from all other cataloged files. The user ID and the file name form part of the catalog entry for this file.

The user himself must specify the names for his files, and can change them as desired.

Format of the complete file name (path name)

```
:catid:$userid.filename
```

Files which are not cataloged under the user ID and in the standard pubset of the current job must be addressed in the above form (see format F2 for an exception). The total length must not exceed 54 characters.

:catid:	A catalog identifier consists of one character enclosed by two colons.
userid	A user ID consists of a maximum of 8 characters.
filename	A file name can be either fully or partially qualified.

Possible specifications:

:catid:$userid.filename
:catid:filename
$userid.filename
filename

Fully qualified file names

A fully qualified file name is the name as it stands in the catalog entry. It precisely identifies this catalog entry.
A catalog entry, in turn, provides a definition of either exactly one file or of exactly one file generation group (see section 2.1.6, "File Generation Groups").

Fully qualified file names can have the following formats:

Format F1 (format of the file name in the catalog entry)

```
name1[.name2]...[({absgen})]
                  {relgen}
```

This format designates a file under one user ID and in a pubset. The overall length must not exceed 41 characters.

The form of the subnames is as follows:

name1/name2 Simple names consisting of:

- At least 1 letter (the name in the catalog entry contains uppercase letters only; however, a file name can also be entered in lowercase letters when input is from the terminal).
- the digits 0...9
- the special characters - (hyphen), @, #, $.

  The 1st character of a simple name must not be a special character; the hyphen must not be its last character. The file name must not consist exclusively of numeric characters.

absgen An absolute generation number *nnnn, where:

$1 \leq nnnn \leq 9999$

(leading zeros can be omitted).

absgen identifies the file generation to other generations of the group.

relgen A relative generation number

$\left\{\begin{matrix}+\\ \\ -\end{matrix}\right\}nn$

where: $0 \leq nn \leq 99$

(leading zeros can be omitted).

relgen refers to the base value (see section 2.1.6).
It is derived as follows:
relgen = absgen - base value.
relgen identifies the file generation to the other generations of the group.

Format F2

$name1

This format designates a file under the standard ID to be generated (DEFLUID parameter at system generation time). Examples of these files are: language processors, file editors.

The overall length must not exceed 44 characters and the name must not contain a period.

Examples of fully qualified file names:

```
$RNO.TEXT.EXAMPLE
GENERATIONGROUP(*20)
GENERATIONGROUP(-5)
```

Partially qualified file names

A partially qualified file name can identify more than one catalog entry. Partially qualified file names can be used only to reference files (or file generation groups) which have already been cataloged; they are permitted only in certain commands/macros (e.g. ERASE, FSTATUS, PRINT, PUNCH).

Partially qualified file names have the following format:

```
name1.[name2.]...
```

The maximum length is 39 characters.

The form of the subname is the same as for fully qualified file names.

Examples of partially qualified file names:

```
$RNO.
$TSOS.
PRIM.FORTRAN.
```

Allocation of file names

When setting up a file, a fully qualified file name must always be specified. Where there are a number of logically associated files, the file names can be selected in such a way that, at the same time,

- the logical connection is apparent

  and

- it is possible to obtain, for instance, status information on all files or a printout of all files by means of a single command.

Example:

Given the following files:

SRCE.B.C SRCE.B.D SRCE.E	These files could contain source programs.
OBJ.Y OBJ.Y.Z	These files could contain object modules.
ENCYCLOPAEDIA	This could be a text file.

In this case, the command
PRINT SRCE.
will generate a printout of the files
SRCE.B.C
SRCE.B.D
SRCE.E
on the printer.

The command
FSTAT OBJ.,ALL
will cause status information for the files
OBJ.Y and
OBJ.Y.Z
to be displayed on the terminal (SYSOUT).

Fig. 2-1 illustrates which files are referenced when a partially qualified file name is specified.

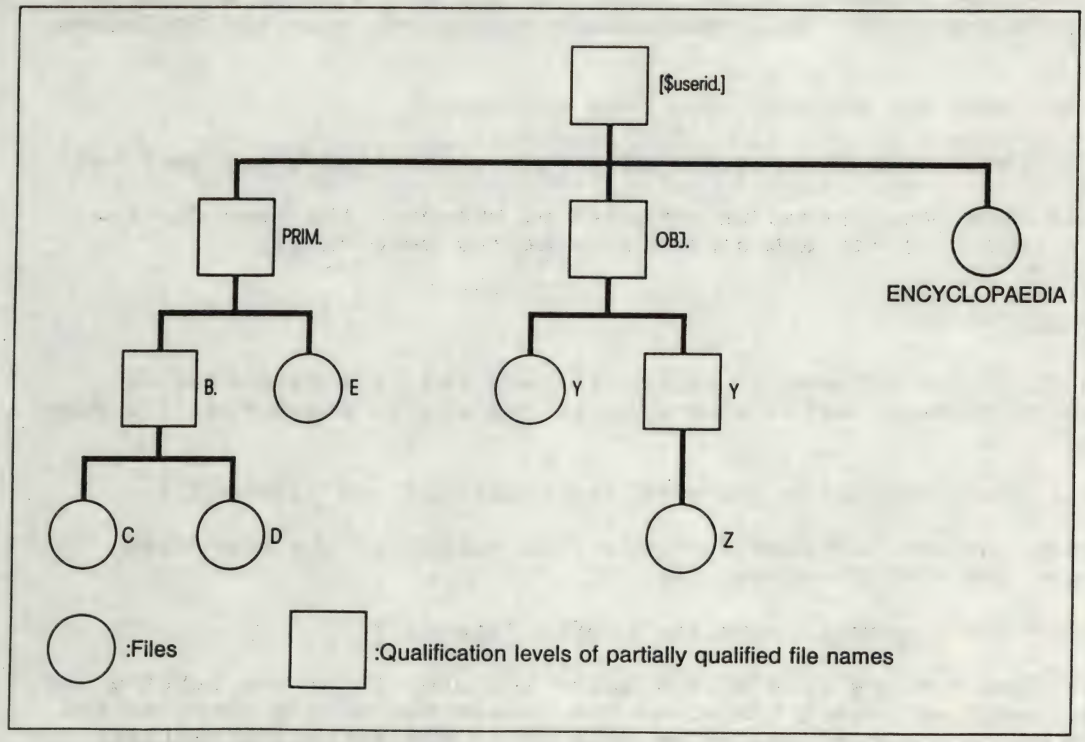


Fig. 2-1: Relationship between files and file names

The following should be noted with regard to the allocation of file names:

Existing (fully qualified) file names can, after further qualification, be used as file names for new files, without having to alter the names of the existing files.

Example:

The name SRCE.E is further qualified to form the name SRCE.E.ADDITN, which could be the name of a file containing additional source program sections. SRCE.E continues to identify the same file as before.

Changing file names

File names can be changed by means of the CATALOG command.

Example:

CATALOG SRCE.E,SRCE.FIRST-PROGRAM,STATE=UPDATE

The name of the file SRCE.E is changed into SRCE.FIRST-PROGRAM.

With disk files (i.e. files on either public or private disks), a file name can be changed to any other (fully qualified) file name, with the following exceptions:

- The file name must not already have been allocated.
- The names of individual generations of a file generation group must not be changed.
  In this case, the name change is effected by changing the name for the group entry (and thus for all generations at the same time).

Retrieving files

To retrieve a file, the DMS has to search for and read the file catalog entry. The search process varies according to the way in which the file name was specified:

- The name was specified using the prefixes ":catid:" and "$userid.".

  In this case, the DMS searches for this file solely in the specified TSOSCAT under the specified user ID.

- The file name was specified with the prefix ":catid:".

  The DMS searches for the file solely under the user ID of the calling job and on the specified pubset (provided the caller has a JOIN entry in the specified pubset). In the case of an OPEN macro the following applies: if the file cannot be found under this user ID, a second read attempt is made under the default ID defined by the system administrator (class 2 option DEFLUID) by specifying OPTION=GLODEF in the FCB macro.

- The file name was specified with the prefix "$userid.".

  The DMS searches in the public volume set assigned to the user ($userid).

- The file name is specified without a prefix.

  The DMS searches for the file under the user ID of the calling job and within the pubset assigned to this user ID, unless OPTION=GLODEF was specified in the FCB macro.

Special cases:

- In order to reference files under the default ID defined by the system administrator (e.g. EDOR, SORT, ASSEMB) it is sufficient to prefix the file name with the character "$".

- Whenever an EXECUTE, CALL, DO, ENTER or LOAD command is entered and the file name is not explicitly specified with a user ID, the file is always sought in a second user ID (default ID).

Example 1:

User ID	$USER
Standard catalog ID	C
DEFLUID	TSOS

Call via /EXEC, i.e. a second read attempt is made.

PVS :A:
$USER.EDOR

PVS :C:
$USER.EDT $TSOS.EDOR

EXEC EDOR	File $TSOS.EDOR is found on public volume set C
EXEC $EDOR	File $TSOS.EDOR is found on public volume set C
EXEC :A:EDOR	File $USER.EDOR is found on public volume set A

Example 2:

The following users are entered in a public volume set:

- TSOS
- DOE
- SMITH

Class 2 option (DEFLUID): TSOS

The following files are cataloged for these users:

TSOS	DOE	SMITH
ASSEMB FOR1 ACCOUNT	PAYROLL FOR1 ALGOL	INVENTORY PAYROLL

Files

The table below illustrates how user references can affect files:

File referenced by DOE	Action
PAYROLL	DOE's PAYROLL file accessed
$SMITH.PAYROLL	SMITH's PAYROLL file accessed, if SHARE=YES
FOR1	DOE's FOR1 file accessed
$FOR1	System's FORTRAN compiler accessed
$TSOS.ASSEMB	System's Assembler accessed
$SMITH.ACCOUNT	Error - SMITH has no file called ACCOUNT
$ALGOL	Error - The file is not in the system administrator's catalog.

2.1.3 Setting Up and Erasing Files

Setting up files

The commands

CATALOG (macro CATAL)
FILE (macro FILE)

are used to set up a file.

The CATALOG command is used to create a catalog entry for a file and to define protection attributes for this file, e.g. passwords for read and write access.

However, the technical characteristics of a file set up in this manner are still not defined; neither is any storage space allocated to this file.

When the FILE command is used, on the other hand, a catalog entry can be created and storage space reserved. The protection attributes of such a file are set to default values; they can be changed by means of a subsequent CATALOG command.

Notes:

Catalog entries for a user ID are created by the DMS in the order in which CATALOG and FILE commands/macros occur.

When a file is opened and when any other action that necessitates accessing a file's catalog entry takes place, the DMS searches the catalog entries for the user ID concerned, one after the other until the required entry is found.

If many files have been cataloged for this user ID, and if many other entries precede the catalog entry required, then searching the catalog is a very time-consuming activity.

To minimize the time required for the search, the following rules should be observed when setting up a file:

- Catalog as few files as possible for one user ID (e.g. approximately 60 files).
- Files which are used most frequently should be cataloged before other files.

Erasing files

The ERASE command is used for erasing files.

A file can be erased in the following ways:

- The file is removed from the system and its storage space released. (This is the default function of the ERASE command.)
  The contents of the file can optionally be destroyed before the space is released (overwriting with binary zeros).

- The storage space is released but the file remains known to the system (i.e. it remains cataloged). Here too, the contents of the file can be destroyed before the space is released.

- The file remains known to the system and the storage space remains allocated, but the contents of the file are no longer accessible (the file is "logically deleted").

- The file is removed from the system; its contents, however, remain unchanged and the storage space remains allocated.
  Only those files located on private volumes (tape or disk) can be erased in this way).

Note:

In a job, only those files belonging to the user ID under which the job is running can be erased. (Exception: system administrator.)

2.1.4 Relationship between Files and Programs in a Job

Relationship based on file names

A program which has to process a particular file can address that file using its fully qualified file name.
This procedure has the following disadvantages:

- The file name must be defined when the program is written.

- If several different files are to be processed in consecutive program runs, the file names of both the file just processed and the next one to be processed must be modified in the catalog after every run.

These disadvantages can be avoided by using a file link name in place of the actual file name for linking a file with a program in a job.

Relationship based on file link names

A program can reference every file it is to process by a file link name.

Which file is associated with a file link name in a particular case can be defined separately for every program run, so that the file names concerned remain unchanged in the catalog.

The link between a file and a file link name applies only to the job in which it was defined.

The FILE command is used to define file link names (operands "filename" and "LINK=linkname"). In order to change an existing file link name, the CHANGE command is used.

The link between a file and a file link name for each job is recorded internally in the associated Task File Table (TFT).

If a program wants to open a file during a job and specifies only the file link name, the DMS establishes the relationship to the file for which the file link name specified was defined.
(For further details see section 5.1, "Opening Files".)

It should be noted that software products use predefined file link names; these can be found in the appropriate reference manuals.

Example:

This example illustrates the expediency of file link names.

A file DATEI1 is to be sorted using the software product SORT. After sorting, the file is to be edited according to certain criteria and output on the printer by the user program LISTE.

The software product SORT employs the following file link names:

SORTIN for the input file
SORTOUT for the output file.

The user program LISTE uses the file link name LISTEIN for the input file. The problem described above can then be solved using the following command sequence:

```
/LOGON XYZ
/FILE DATEI1,LINK=SORTIN                 ┐ The FILE commands establish the
/FILE DATEI1.SORTIERT,LINK=SORTOUT       ├ link with the file link names.
/EXEC SORT                               ┘
   .
   .     (instructions to SORT)
   .
/CHANGE SORTOUT,LISTEIN                  ┐ The CHANGE command changes the
EXEC LISTE                               ├ file link name SORTOUT to
/LOGOFF                                  ┘ LISTEIN.
```

2.1.5 File Protection

Defining protection attributes

The DMS offers a number of facilities for protecting files against inadvertent destruction or unauthorized access. For any of his files a user can restrict (by means of the CATALOG command) the number of co-users and also the type of access allowed. The owner of a file is considered to be the user under whose user ID the file is cataloged.
In particular, the owner of a file may specify the following:

- The system administrator can restrict access to a pubset so that only those users explicitly authorized to access the pubset (JOIN entry) can do so.
- Access to the file can be restricted to the owner's jobs, or may be permitted for all user jobs (in which case the file is referred to as being shareable).
- The file can be provided with passwords for read access, write access and execution (if the file contains load modules, object modules or procedures). The DMS then permits only those jobs having the appropriate passwords to access the file.
- A retention period can be specified for the file, during which the DMS does not allow the file to be overwritten, extended or erased. The file can be used only as an input file during this period.
- All write access to the file can be prohibited, in which case the DMS prevents the file being overwritten, extended or erased. The file can be used as an input file only.

Notes:

If no protection attributes are specified when a file is set up, the system incorporates the following default values:

- only the owner can access the file
- no passwords
- no retention period
- writing permitted.

The owner of a file can change the protection attributes at any time (by means of the CATALOG command).

If a file is provided with passwords, these must be specified (by means of the PASSWORD command) before the catalog entry is changed.

Example:

User DOE updates the following catalog entries:

```
(1)  /CATALOG DATEI1,SHARE=NO,ACCESS=READ,STATE=UPDATE
(2)  /CATALOG DATEI2,SHARE=YES,ACCESS=READ,STATE=UPDATE
(3)  /CATALOG DATEI3,SHARE=YES,WRPASS=C'007',STATE=UPDATE
(4)  /CATALOG DATEI4,SHARE=YES,RDPASS=C'1111',STATE=UPDATE
(5)  /CATALOG DATEI5,SHARE=YES,RDPASS=C'1111',EXPASS=C'2222',STATE=UPDATE
```

(1):

Only user DOE can access the file DATEI1. Not even DOE, the owner, can write to the file while write access is forbidden (ACCESS=READ).

(2):

All users of the system can read $DOE.DATEI2.

(3):

All users of the system can read $DOE.DATEI3. Only those users who specify the password C'007' have write access.

(4):

Only those users who specify the password C'1111' can read or write in $DOE.DATEI4.

(5):

Users who omit both passwords can neither read nor execute the file.
If a user specifies the execution password, he can execute the file, but not copy it.
If he also specifies the read password, he is able to read the file and even overwrite it.

Access to password-protected files

If a job wishes to access a file protected by passwords, it must specify the passwords necessary for that access, using one of the two following methods:

- with a PASSWORD command (any number of passwords).
  The passwords given in this way are placed in the job's password table, and do not therefore need to be entered anew for every access.

- With the FCB macro, used when the file is opened (only one password permitted).

Passwords have the following order of priority:

- execute password (highest priority)
- read password
- write password (lowest priority).

This means that each password also has the function of the lower priority passwords. This extended protection is rendered ineffective if the password with the lower priority is explicitly specified. The following table shows which passwords should be specified to obtain the access required.

Key:

E = Execute the file contents (EXEC, LOAD, DO, ENTER commands).

R = Read the file (OPEN mode INPUT).

W = Write the file or change the catalog entry (OPEN modes OUTPUT, OUTIN, INOUT; CATALOG command; FILE command with SPACE operand.

The gray shaded areas in the upper part of the table indicate that the password in the left-hand column of the table is cataloged.

	Desired access																				
	E	R	W	E	R	W	E	R	W	E	R	W	E	R	W	E	R	W	E	R	W
Execution password cataloged																					
Read password cataloged																					
Write password cataloged																					
Execution password required	+	+	+							+			+	+					+		
Read password required					+	+					+	+					+			+	
Write password required									+						+			+			+

Table 2-1: Password specification requirements in relation to the desired type of access

System administrator and file protection

The system administrator is the only user who has privileges over other users with regard to accessing files.

The system administrator can obtain information relating to the protection attributes of any file entered in the system catalog, and if necessary modify these attributes. This facility is necessary to ensure the smooth running of the system. For example, if a user should forget the passwords for his file, the system administrator can read out the passwords, or, in the case of systems with coded passwords, can delete the passwords so that the file once more becomes accessible for the user.

As a rule, the system administrator has access to every file entered in the system catalog, regardless of the owner.

Protection mechanisms for multiple jobs simultaneously accessing one file

In addition to file protection by means of catalog attributes, the DMS also provides mechanisms which prevent the mutual interference of jobs wishing to process the same file simultaneously.

As a rule, the DMS does not permit more than one job to operate with the same file. The following are the exceptions to this rule:

- Several jobs can simultaneously access one file if each individual file wishes only to read it.
  (The protection attributes of the file are, of course, still observed.)

- In the case of the access methods UPAM and ISAM (see chapter 6, "Access Methods"), the DMS provides the "shared file update" function, which allows multiple jobs to read into and write from a file simultaneously. To prevent interference between jobs, parts of a file can be locked temporarily by a job. These parts cannot be accessed by other jobs until they are unlocked by the locking job (mutual exclusion).

2.1.6 File Generation Groups (FGG)

General

A file generation group (FGG) is a collection of cataloged files -- the generations -- with the following characteristics:

- The order in which the individual generations are created (the chronological sequence) remains known to the system. This is effected by means of the allocation of a consecutive number (the absolute generation number) to each one.
  Each absolute generation number corresponds exactly to a position in the chronological sequence; the "less than/greater than" relationship between two absolute generation numbers represents the chronological relationship "older/newer" between generations.

  Exception:

  If the absolute generation number of a generation is 9999, the successor generation is given absolute generation number 1.

Example:

File generation group structure

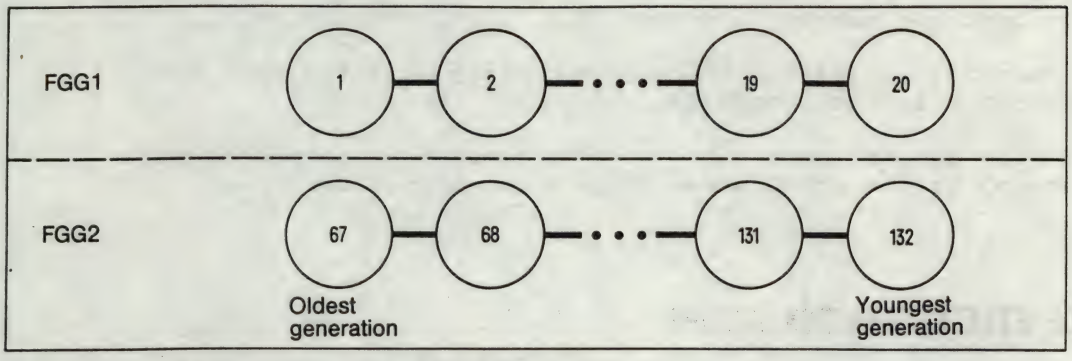


Fig. 2-2: File generation group structure

- Each generation can be addressed by a relative generation number. The user himself can select the point of reference for relative numbering, but should first ensure that the generation selected as the point of reference is cataloged (see also the subsection "Changing the relationship between relative and absolute generation numbers" below).

Example:

Relationship between relative and absolute generation numbers

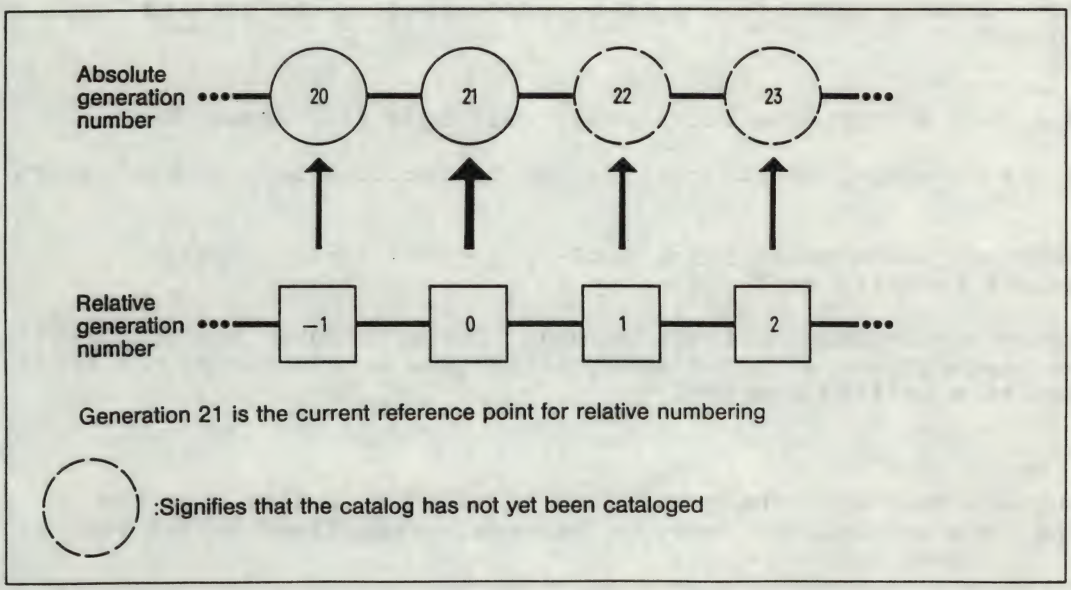


Fig. 2-3: Relationship between relative and absolute generation numbers

- Catalog characteristics, such as shareability, read or write access, file passwords etc. apply equally to all generations.
  They are characteristics for the file generation group as a whole.

File generation groups are particularly suitable for regularly recurring jobs, such as customer master file updating, invoicing etc.

The user of relative generation numbers dispenses with the need to formulate the job prior to processing.

In addition, file generation groups offer a convenient method of data saving, according to the grandfather-father-son principle.

PROCESSING OF FILE GENERATION GROUPS

Setting up file generation groups

The system manages file generation groups by means of a catalog entry (the group entry), which differs from a normal entry for a file, and which can be created only by means of a CATALOG command/macro.

In order to create the group entry, the CATALOG command/macro must contain the following:

- the group name (operand "filename1") and
- the maximum number of generations to be maintained in the catalog (operand GEN=)

The group name must differ from all already cataloged file names.

In addition, the following specifications can be included in the group entry regarding:

- file protection (operands ACCESS=, SHARE=, RDPASS= etc.). These specifications normally apply globally to all generations.

  However, when working with private volumes (EXPORT/IMPORT) the user must ensure that generations with different protection attributes do not arise as a result of a CATALOG command.

  Exception:

  The catalog attributes of the group were modified at a time when one or more groups were unavailable (message DMS06A9: GENERATIONS OF THE FGG MISSING).

  If the user wishes to restore the normal status, he must issue a further CATALOG command/macro in order to update the FGG with the desired catalog attributes. This then causes the catalog entries of all associated generations to be updated.

  The determining factors for file access to an FGG are the attributes defined in the catalog entry of the FGG.

- the procedure to be adopted when the maximum number of generations allowed in the catalog has been reached (operand DISP=). It is possible to specify that, in this case, the oldest generation is to be deleted upon cataloging the newest (either with or without reusing the private volumes used by the deleted generation), or that all generations are to be simultaneously deleted or retained.

Upon recataloging the group entry of existing file generation groups on private volumes, one of the following specifications must additionally be made, regarding:

- the chronological order (for tapes). Specify

  - the oldest generation to be held in the catalog (operand FIRST=), and

  - the base generation to be used as a reference point for relative numbering (operand BASE=).

  Otherwise, the group entry is to be generated in the same way as any new catalog entry (operands STATE=NEW, GEN=).

- the device and the volume serial number (for disks). The operands STATE=FOREIGN, DEVICE=, VOLUME= are required in order to transfer the group entry from the F1 label to the catalog.

Example 1:

A file generation group GROUP1 is to be set up on public volumes.

The maximum number of generations to be held in the catalog is to be set at 5. When this number is reached, the oldest generation is to be deleted. The file generation group is to be shareable.

These attributes are defined in the group entry by the following command:

```
/CATALOG GROUP1,GEN=5,DISP=CYCLE,SHARE=YES,STATE=NEW
```

Example 2:

A file generation group GROUP3 on private disks is to be recataloged. In this case, the group entry must be read from the F1 label and placed in the catalog.

This is achieved by means of the following command:

```
/CATALOG GROUP3,STATE=FOREIGN,DEVICE=D3455,VOLUME=DSK011
```

Creating generations

Generations of an FGG are always created using the FILE command/macro or, if only the catalog entry is to be created, with the CATALOG command.
A generation name with absolute or relative generation number must be specified.

Note:

When creating generations for the same FGG, the generation numbers used in the generation name must satisfy the following conditions:

Absolute generation numbers	Relative generation numbers
must not be used multiply.	must not be used multiply within one job. For an exception, refer to "Changing the relationship between relative and absolute generation numbers" below.
must be greater than zero.	must be specified with the sign "+" or "-".

The resultant absolute generation numbers must always be in ascending sequence, and there must always be a difference of 1 between two absolute generation numbers created one immediately after the other, i.e. there must be no gaps.

(See the CATALOG and FILE commands for a description of the format.)

Example 3:

Creating generations of file generation group GROUP1 on public volumes

```
/FILE GROUP1(+1),LINK=OUT1
/FILE GROUP1(+2),LINK=OUT2
/FILE GROUP1(+4),LINK=OUT3
%DMS06C7 USER ATTEMPTED TO PROCESS A GENERATION WITH AN INCORRECT
GENERATION NUMBER. COMMAND TERMINATED.
(Generations must be created consecutively without gaps, hence)
/FILE GROUP1(+3),LINK=OUT3
```

Notes:

All generations of an FGG on public volumes must be on the same pubset, i.e. all generations of an FGG have the same catalog ID.

The following restrictions apply for:

- file generation groups stored on private volumes.
  If a file generation group stored on private volumes is known to a system, it must be cataloged in its entirety on one pubset, i.e. including all existing generations. Such an FGG can also be cataloged in more than one pubset; however, in this case it must be cataloged in its entirety in every pubset.

- file generation groups stored on public volumes.
  These FGGs must be contained in their entirety on one pubset, i.e. including all existing generations.

Deleting generations

The same applies when deleting generations as when creating them, in that no gaps must occur in the sequence of the absolute generation numbers. Thus, starting from any arbitrarily selected generation, only all older or all newer generations can be deleted (operand POS= in the ERASE command/macro).

If the previous base generation for relative generation numbers is deleted by an erase operation, the generation specified in the ERASE command/macro becomes the new base generation.

An erase operation always updates at least one of the fields in the group entry, which contain the absolute generation numbers of the oldest and newest generations entered in the catalog. This is not the case when only the catalog entries of generations on private volumes are deleted.

Note:

Gaps may occur in a file generation group when using the ERASE command with the operands CATALOG and VOLUME=vsn for private disks; see also the note for the IMPORT command.

Example 4:

File generation group GROUP5 has the following form:

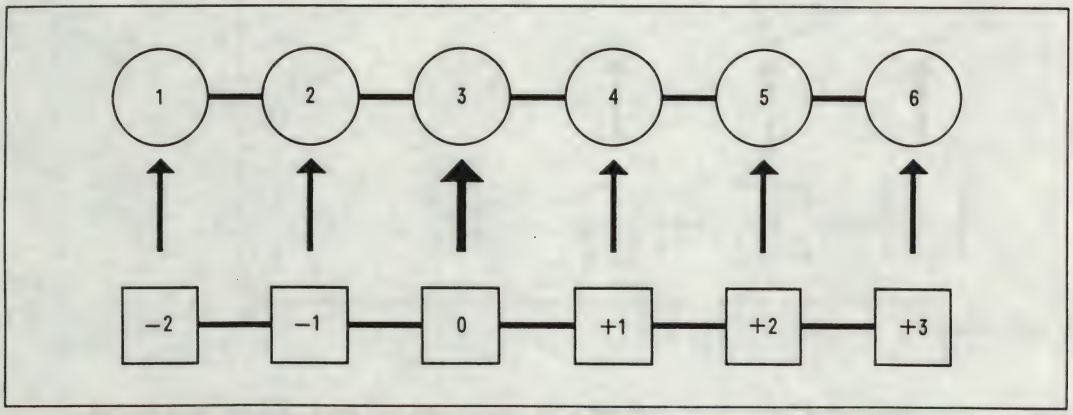


Fig. 2-4: Form of GROUP5

Taking generation 3 as the base, all older generations are to be deleted. This is done using the following command:

```
ERASE GROUP5(+0),POS=BEFORE
```

After execution of this command, GROUP5 has the following form:

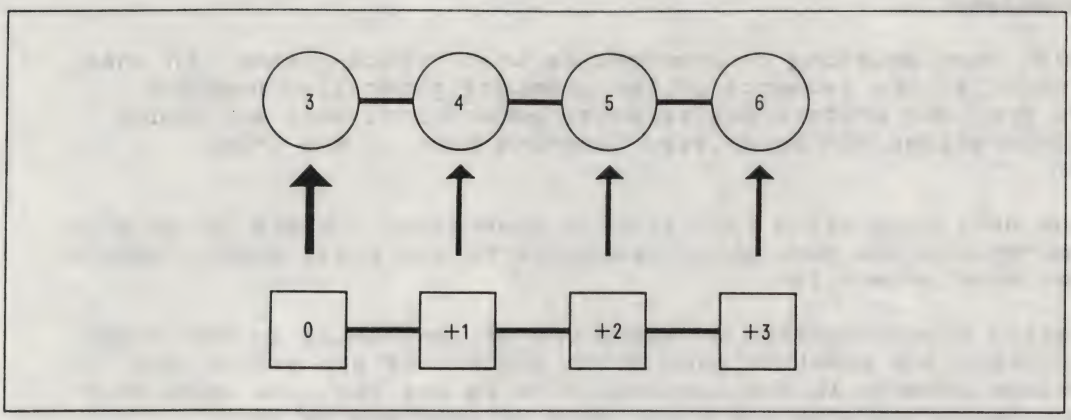


Fig. 2-5: Form of GROUP5

If generation 3 is also to be deleted, this can be achieved by the following command:

/ERASE GROUP5(*4),POS=BEFORE

After execution of this command, GROUP5 has the following form:

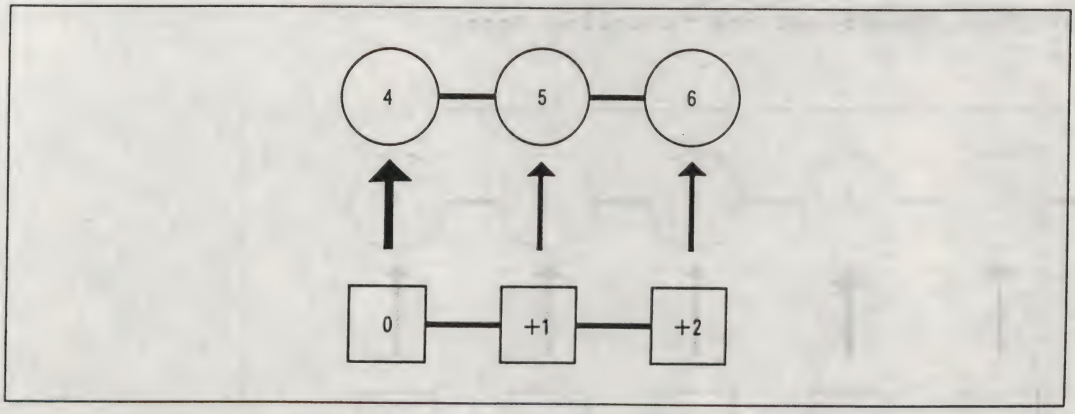


Fig. 2-6: Form of GROUP5

Since the previous base generation for relative numbering has been deleted, the relationship between relative and absolute generation numbers had to be redefined.

Changing the relationship between relative and absolute generation numbers

It is possible to change the relationship between relative and absolute generation numbers during a job by means of a CATALOG command (operand BASE=). The group name must be specified, as well as the generation which is to serve as the new reference point for relative generation numbers. This generation is identified by an absolute or relative generation number. Only a generation which has already been cataloged may be selected as the new reference point for relativization.

Note:

In this case, the reference point for relative generation numbers is the newest generation at this point in time, and may therefore differ from the newest generation at the time of job initiation. This always applies when new generations have been created during the course of the current job.

Example 5:

File generation group GROUP6 has the following form:

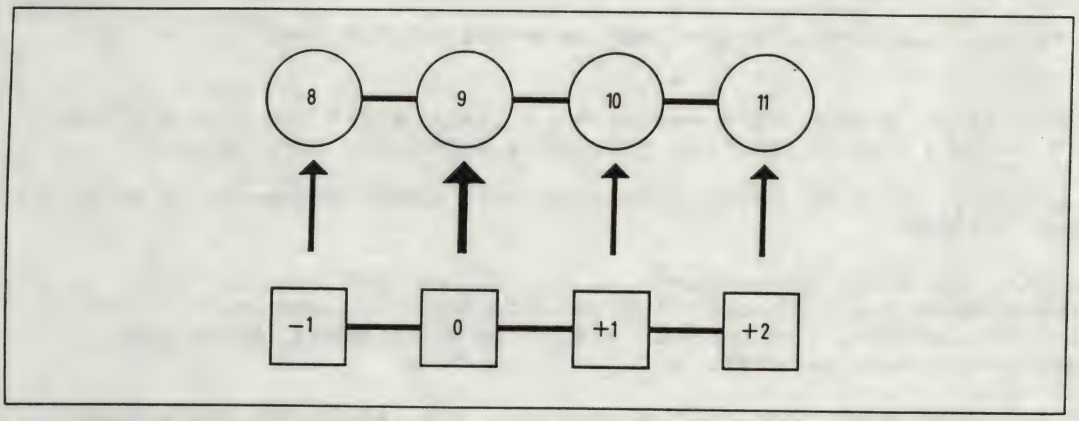


Fig. 2-7: Form of GROUP6

The command

/CATALOG GROUP6,BASE=-2,STATE=UPDATE

defines the following relationship:

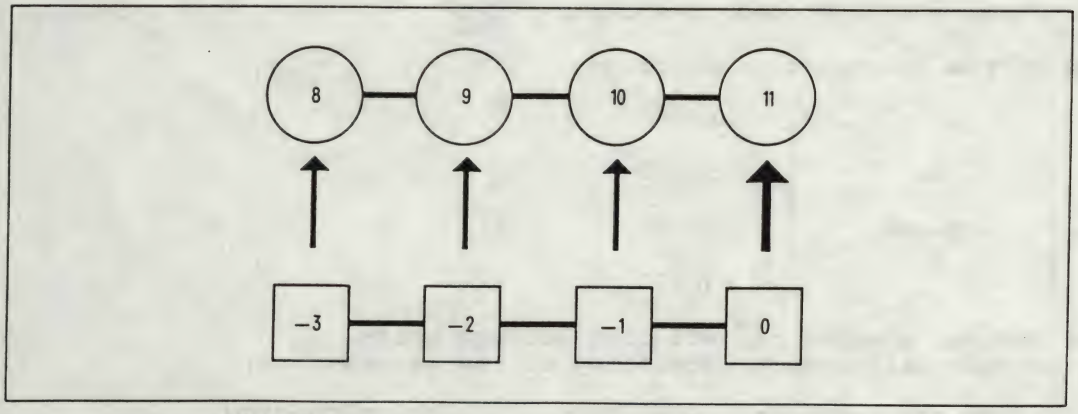


Fig. 2-8: Form of GROUP6

Volumes for file generation groups

File generation groups can be created and processed on public and private volumes.

Notes:

- The mixed use of public disks and tapes is permitted.
- The use of private disks for an FGG precludes the use of other volumes for that FGG. Device and volume specifications must be made when creating the group entry and the generations (operands DEVICE= and VOLUME= in the CATALOG command).

  In extreme cases, a separate volume can be provided for each generation.
- If a file generation group which exists on private disks but has not been cataloged is to be recataloged, the following procedure must be used:
  - The group entry is to be created as described under "Setting up file generation groups".

    Subsequently, at least those generations which occur between the oldest generation maintained in the catalog and the base generation for relative numbering (inclusive) are to be cataloged, using one FILE command for each generation.

    The FILE commands must contain the operands STATE=FOREIGN and DEVICE= and VOLUME=.
  - The IMPORT command provides a further facility for cataloging the file generation group, but certain special conditions must be observed. See also the IMPORT command.

Example 6:

All generations of file generation group GROUP3 which are contained on private disks are to be recataloged (see Example 2).

GROUP3 consists of the following generations:

Generation 1	on volume	DSK006
Generation 2		DSK003
Generation 3		DSK150
Generation 4		DSK150

The volumes are already mounted on 3455 Disk Storage Units.
The following commands will effect cataloging of the generations:

```
/FILE GROUP3(*1),LINK=DZ1,DEVICE=D3455,VOLUME=DSK006,STATE=FOREIGN
/FILE GROUP3(*2),LINK=DZ2,DEVICE=D3455,VOLUME=DSK003,STATE=FOREIGN
/FILE GROUP3(*3),LINK=DZ3,DEVICE=D3455,VOLUME=DSK150,STATE=FOREIGN
/FILE GROUP3(*4),LINK=DZ4,DEVICE=D3455,VOLUME=DSK150,STATE=FOREIGN
```

Reserving file generation groups

A file generation group (including all generations) can be reserved for a job by specifying the group name and the operand EX in the SECURE command. The reservation does not cover the volumes; it merely prevents another job from accessing the FGG or one of its generations (by means of a lock in the File Lock Table, FLT).

Note:

Simultaneous accessing of one FGG by multiple jobs can lead to unpredictable results.
It is therefore always advisable to make use of the reservation facility in cases where simultaneous accessing of one FGG by multiple jobs cannot be ruled out.

Example 7:

File generation group GROUP9 is being processed by job 1 and has the following form:

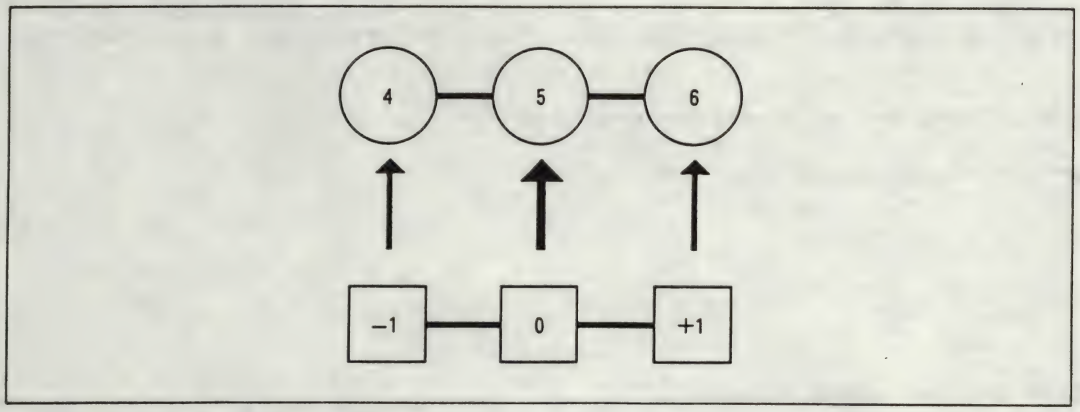


Fig. 2-9: Form of GROUP9

At this moment, job 2 is started. This creates generation 7, erases generation 4 and is then terminated.

GROUP9 now has the following form:

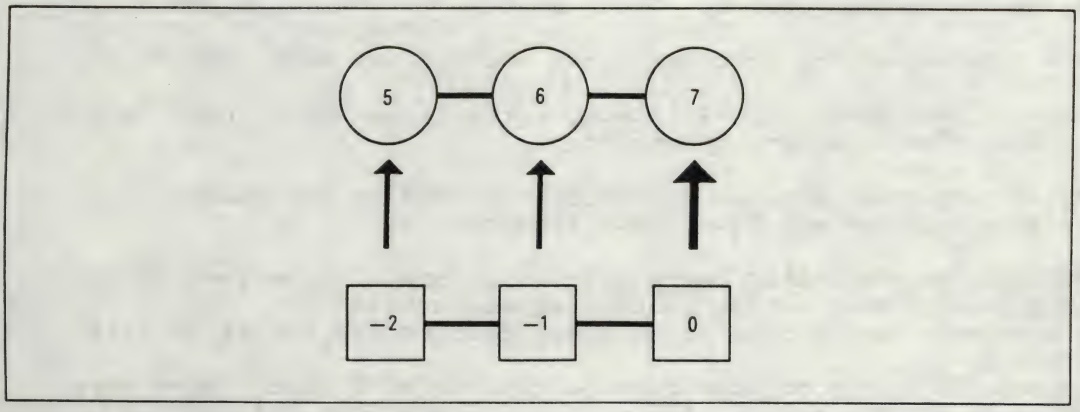


Fig. 2-10: Form of GROUP9

The use of relative generation numbers will always result in the wrong generation being accessed.

If GROUP9(+0) is specified, job 1 will now address generation 7 and not the desired generation 5.

Even if absolute generation numbers are used, an error will occur as soon as the extinct generation 4 is referenced.

Non-executability of generations

The generations of an FGG are not executable files. Thus, there is no point in using them to store

- ENTER/DO procedures or
- load and object modules

Support of file generation groups by utility routines or software products

File generation groups are not supported by all utility routines or software products.

Non-supporting routines and software products are:

AUPMAIN (for system administrator only)
LMS
PRSERVE
TSOSLNK.

2.1.7 Restoration of Corrupted Files

If processing of a file is terminated abnormally (the job processing the file, referred to below as the "processing job", is terminated, e.g. due to a system failure, or it is deactivated until the end of the session, e.g. due to memory saturation), the file may be inaccessible for subsequent processing.

This is due to the following reasons:

- The file lock imposed by the processing job has not yet been removed.
- The file contents and parts of the catalog entry (e.g. the pointer to the last PAM block in the file) are inconsistent.
- With ISAM files there may also be disparities in the relationships between file data section and file index section.

By using the VERIFY command (VERIF macro), it is possible to restore files corrupted in this way so that normal processing may continue.
In the program a branch can be made to an error routine via the EXLST exit OPENC (cf. section 5.4).
The FSTATUS command (FSTAT macro) can be used to determine which files have not been closed, e.g. as the result of a system failure (see the operand STATE=NOCLOS).

Restoring corrupted files

- The file must be unlocked if it is still locked (remove the entry for this file in the File Lock Table, FLT).

  However, a check is made as to whether the lock is not in fact still required.

  The lock is removed only if

  - the file was locked by a SECURE command and the processing job is in a constantly inactive state or

  - the file is a tape file and the processing job is in a constantly inactive state.

  With disk files that were not locked by a SECURE command, only the system administrator can remove a lock.

- Disk files which have already been unlocked are handled differently according to the access method used to create them:

  - In the case of an SAM file, the pointer in the catalog entry is changed so as to indicate the last PAM block of this file (with normal file processing, this pointer is not set until the file is closed).

    The file can then be reopened (also in EXTEND mode, if required).

    It now also becomes possible once more to access the last records placed in the file. However, records which were being processed in the buffer immediately prior to job termination might be lost; see also "Limitations on file restoration" below).

  - In the case of an ISAM file, all records which can still be restored are written to an ISAM restoration file with new key blocks.

    It is possible to specify the file name under which the restored file is to be stored (operand "filename2" in the VERIFY command/VERIF macro).

    If this file is contained on private disks, it must already have been provided with sufficient storage space. If a user-supplied file is used for restoration, the corrupted file is not erased.

    If no second file name is specified, the corrupted file is restored in a work file (on public volumes).

    If the corrupted file is on

    * private volumes, it is overwritten by the contents of the work file.

      The work file is subsequently deleted (and its data is destroyed for file security reasons).

    * public volumes, it is deleted (without data destruction, in order to save time) and the work file receives the name of the original file.

- In the case of a UPAM file, a privileged close operation is performed. If necessary and possible, the file is unlocked. Subsequently the pointer is updated so as to point to the last PAM block (LPP) of this file. This ensures that none of the blocks containing data are released in the event of a subsequent FILE command/macro with a negative SPACE specification.

  The restoration of a UPAM file and simultaneous updating of the LPP are initiated via the VERIFY command (VERIF macro) and the operand REPAIR=ABS.

  Note:
  This function requires that a file is opened and then read backwards until the current PAM page is reached. Consequently, it can be very time-consuming in the case of large files.

Restoring corrupted files under a foreign user ID

If a foreign user ID is specified in a path name, the file must be cataloged as shareable.

If no other file was specified, a scratch file is created under the user's own ID. After restoration, this scratch file is copied into the original file and deleted by means of ERASE,DESTROY. If a third file is required, it too is created under the caller's user ID. As the scratch files are initialized in the disk set containing the file to be restored, the caller must have access authorization for the disk set of the specified file.

Limitations on file restoration

- As a result of the buffering in main memory of logical records from SAM and ISAM files, changes made immediately prior to termination of file processing might be lost.

  With ISAM files, this problem can be overcome by immediately writing back to the volume (operand WROUT=YES in the FILE command). It should be noted, however, that this process increases overall time expenditure.

- When restoring ISAM files, if any duplicate records are involved (i.e. records with identical keys and identical data), only one such record is included in the restoration file in each case.

  This restriction is required to permit the restoration of ISAM files whose processing was terminated during a block splitting operation.

- ISAM files with separate volumes for file index section and file data section can be restored only if the size of the data blocks is one PAM block.

- If a data block of an ISAM file cannot be restored (e.g. due to the accidental destruction of internal file pointers, or because an updated multi-block buffer could be only partially written back to the volume), the block is written to a special PAM file.
  This PAM file is available to the user after completion of reconstruction for his own restoration needs.

  The file name is as follows:

  S.filename-corrupted-file.REPAIR

- If a corrupted ISAM file contains multiple records with identical keys (but different data), the sequence of these records in the restoration file may differ from the original order.

- With the restoration of ISAM files, no free space is left in the data blocks for subsequent additions (the value of the PAD operand in the FILE command is ignored).

- The time factor involved in the restoration of ISAM files should not be ignored.

  It can increase considerably in the case of files on private volumes if the user does not supply a restoration file (already provided with sufficient storage space). (See the operand "filename2" in the VERIFY command/VERIF macro.)

2.2 VOLUMES

2.2.1 Public and Private Volumes

In BS2000, there are two different types of volume:

- public volumes
- private volumes.

Only disks can be public volumes. Private volumes are all tapes and all disks not marked as public volumes.
Both types of volume can be used together.

When defining disks, the DMS distinguishes between two different types:

- PVS (public volume sets)

- private disks.

A PVS consists of one or more public volumes. The system itself, together with a number of other files which it needs, is stored on a PVS known as the "home pubset", which is available throughout the session. Other pubsets need not be made available until actually required (as also applies to private volumes).

The DMS assumes responsibility for memory management on disks and guarantees full data protection for all volumes. In this case certain check functions (access authorization, saturation control, etc.) apply only to pubsets and not to private disks.

2.2.2 Space Management (Disks)

- In the case of primary allocation, memory space is allocated to a file for the first time, and a catalog entry is created.

  Whenever possible, the system assigns contiguous space, and arranges even distribution over the public volumes employed.

- With secondary allocation, additional space required by a file is allocated.

  This procedure can be repeated as long as the total amount of space available to the user ID has not been exhausted (see also the next section, "Requesting public and private volumes").

  Dividing a file between several volumes should be avoided if possible.

- Storage space is allocated in the following units:

  - unit (= 3 PAM blocks)
  - packet (= 8 units = 24 PAM blocks)
  - segment (= 8 packets = 64 units = 192 PAM blocks).

  The user can adapt primary and secondary allocation to his particular requirements by way of the SPACE operand in the FILE command.

- If the system is unable to fulfil a request for space, the following occurs:

  - Either the FILE command issued for a primary allocation is rejected, or the request for space is honored in part.

  - In the case of a secondary allocation, control passes to the NOSPACE exit (see the EXLST macro).

- Grounds for the non-allocation of storage space on public volumes:

  - At the time the request was made, saturation of the relevant pubset had reached level 4, or would have done if the allocation requested had been granted.

    The procedure for this eventuality can be defined at system generation time (class 2 option):

    * the request is rejected immediately, or

    * the operator decides whether to accept or reject the allocation request.

    The allocation of storage space for a pubset is controlled via 5 saturation levels. On levels 1 (lowest saturation level) through 3 all allocation requests are accommodated, memory bottlenecks being indicated by messages at the console.

    On saturation level 4, jobs requesting memory space are placed in a queue until a lower saturation level has been reached.

    On level 5 (highest saturation level), only system administrator jobs may request memory space; requests from other jobs are rejected.

    Note:
    Saturation levels 4 and 5 must be avoided for the home pubset (by means of deleting files that are no longer needed or that cannot be restored) in order to avoid endangering a renewed startup.

  - If the request were satisfied, the user would occupy more space on the relevant pubset than is at his disposal.

  - The request exceeds the upper limit on public space available for the user ID under which the job is being executed.

- Grounds for the non-allocation of storage space on private volumes:

  - The capacity of the volume(s) has been exhausted, and thus no further space allocation is possible.

2.2.3 Management of Public and Private Volumes

Requesting public and private volumes

- A request for space on public volumes is effected by means of the FILE command.

  By means of the "catid" specification the user can determine the pubset on which space is to be allocated, but he cannot influence the choice of the disks on which he is allocated space (if the pubset consists of more than one disk). However, he can request information on the success or otherwise of the allocation by using the FSTATUS command/FSTAT macro or the DCAT utility routine.

  Each pubset contains the following information:

  - the users who are authorized to access this pubset are specified; other users have no access authorization;
  - an upper limit is defined for every user with access authorization;
  - those users with access authorization who may exceed this upper limit are specified.

  Users who are not authorized to access a pubset cannot request memory space on this pubset.

  A user with access authorization may request space until he has reached the upper limit defined for him. If he is authorized to exceed this limit, he may continue to request space.

  The operator receives a message informing him that a certain user has exceeded his prescribed limit. If the user has still exceeded his limit at LOGOFF, he receives a message to this effect.

  The storage limit is set by the system administrator. The user can request information on this limit by means of the command /ACCOUNT or /SHOW-USER-ATTRIBUTES.

- Requests for private space or private volumes are effected by a FILE command/macro.

  A private volume or device is made available for the duration of the requirements of the jobs concerned.

  Space on private disks can be requested in any quantity (limited only by the capacity of the volume).

  In addition, the following may be specified:

  - the volume(s) on which space is to be reserved, and
  - which physical blocks are to be reserved.

  (See also the operands VOLUME= and SPACE= in the FILE command.)

Operating modes for private volumes

A private file can be used:

- exclusively by one job.
- jointly by a number of jobs executing simultaneously.

The desired operating mode is specified by the operator.

If a file extends over several private disks, all the volumes must be mounted at OPEN time.

Releasing private volumes

Private volumes and devices are released when one of the following conditions applies to all jobs concerned:

- LOGOFF
- RELEASE
  This command serves to release the devices and volumes allocated to a file.
- SECURE[-RESOURCE-ALLOCATION]
  Specifying the SECURE[-RESOURCE-ALLOCATION] command without operands, or requesting further volumes or devices, means that all those previously reserved are released in order to avoid deadlock situations.
- WHEN
  All devices and volumes are returned to the system, as this command can be used to deactivate a job and place it in the job queue.
- Abnormal termination of job.

Use of private volumes belonging to another computer center

The private disks of one BS2000 computer center can be used in other BS2000 computer centers.

Notes:

- A file on a private disk can be transferred to another system only if the owner's user ID is entered in the other system.
  This applies whether the file is transferred by the owner himself or by the system administrator.
  Exception: If IMPORT with NUSERID=new userid is specified by the system administrator, the old user ID need not still exist (see the "System Administrator's Guide" manual.
  In both cases the file to be transferred is cataloged under the user ID of the owner; the DMS takes the file attributes from the F1 label of the volume, **not** from a FILE command or from an existing catalog entry.

- Transfer to the other system can be effected in the following ways:

  (1) An IMPORT command can be used if more than one file is to be transferred. This command also processes partially qualified file names.

  (2) Transfer of a single file can be effected by means of a FILE command containing the following operands:

  filename
  STATE=FOREIGN
  VOLUME=vsn
  DEVICE=device

  If the file is to be transferred under the system administrator's ID, the file name should be prefixed by the owner's user ID.

  (3) Catalog entries can be saved in the source system and created in the target system by means of the RECAT utility routine.

Information on the use of tapes from other computer centers in a BS2000 computer center can be found in the "DMS Tape Processing" manual.

Pool monitoring (MVDF)

The total number of private disks of a computer center is divided into pools.
The whole file must belong to a single pool and must be located on this pool. In other words, the areas of the file extend over private disks that all belong to the same pool.
The pool affiliation of a disk is entered in the SVL. Formation and administration are performed by means of the PDPOOLS utility routine (see the "System Administrator's Guide" manual).

This arrangement permits sets of files lost as a result of disk crashes to be restored without any problems.

A check is performed at the following stages (arranged in descending order of importance):

- Creation and static extension of a file (FILE command)
- Opening a file (OPEN)
- Dynamic extension of an opened file.

2.2.4 MPVS

MPVS (Multiple Public Volume Set) supports several independent sets of public disks on a single system. These sets are referred to in this manual as "pubsets".

The creation of MPVS represents an attempt to combine the advantages of public and private disks, the most notable improvements being the following:

- Distribution of a large disk pool comprising several disks over a number of small pubsets each consisting of a few disks; this means that the size of a pubset to be restored in the event of an error can be limited without sacrificing the performance advantages of a large paging area.

- Replacement of private disk pools by appropriate pubsets; this enables the advantages offered by the system for processing files on public disks to be used here.

- Enhanced availability; if a processor crashes in a multiprocessor system, its pubsets can be made available in the remaining processors, the entire catalog and data structure being retained.

- Improved data protection via the option of restricting the shareability of a file.

- Reduced restoration times following the destruction of a pubset, thanks to the smaller size.

Several pubsets (max. 36) can exist in one system. Each system contains a home pubset, which is necessary for loading, operating and terminating the system and must be available throughout the session. All other pubsets can be imported and exported as required by the system administrator during the session, with the exception of pubsets containing files required for paging. A pubset can consist of several disks.

The user can create, process and delete files, using all logical DMS functions, on the disks of a pubset. The system recognizes and manages all the disks on a pubset as a single unit.

The pubsets are uniquely identified by the catalog identifier (catid), which is the fourth character in the VSN.

Example:

PUBA01	catalog identifier A
PUBZ04	catalog identifier Z

Each pubset contains a JOIN file and its own TSOSCAT.

Those users who are authorized to access files on a given pubset are specified via their user ID in an entry in the JOIN file:

- If a user is not entered in the JOIN file of a pubset, he cannot access this pubset or shareable files of other users, unless the class 2 option FSHARING was set at system generation time. This option allows access to a pubset under another user ID.

- If the user has an entry with PUBSPACE=0 in the JOIN file, he can access all shareable files (SHARE=YES) but cannot create any files of his own. If PUBSPACE>0 is specified, he can also create files of his own.

2.2.5 Space Accounting

The space management facility provides the job management with data for the creation of accounting records. The data evaluated consists of:

- number of PAM blocks reserved
- duration of reservation.

Further details can be found in the "System Administrator's Guide" manual.

2.2.6 Summary

The previous sections dealt with the concept of volumes in BS2000. The salient characteristics of the various volume types are outlined in Table 2-2.

Readers should note that pubsets will henceforth be the data volumes used in BS2000. The development of private disks will be abandoned.

Volume characteristics	Volume type	
	Pubset MPVS	Private volumes
Storage medium	Disk	Disk; tape
Number of jobs which may use the volume simultaneously	Any number	Disk: 1 (if the disk is exclusively reserved; otherwise any number). Tape: 1
File protection	The DMS guarantees full file protection (cf. section 2.1.5, "File Protection")	
Processing methods allowed	DMS access methods of the operating system; I/O macros of the operating system that process system files; TOS Monitor macros that process TOS system files.	
Labels		For disks, standard labels are supported, but not user labels.
Volume serial number	6-character alphanumeric field; the first three characters must be PUB; the fourth is the catalog identifier (catid) of the pubset.	6-character alphanumeric field; the first three characters must not be PUB.
Volume is mounted	For the home pubset, at the start of the session; other pubsets can be imported by the system administrator as required but must be imported before the pubset is first accessed.	When requested by a job.
Volume is dismounted	The home pubset is retained throughout the session. Other pubsets can be removed by the system administrator when they are no longer required.	Tape devices: on request, or when the job requesting the volume is terminated; disk devices: by the operator, partly on request from the system (new MOUNT message).
Space dynamically acquired during execution of a user program	Yes	Yes
SECURE command required	Not possible	Yes, unless the user wants to risk his job being aborted if the volume is not available (e.g. no suitable device is free).

Table 2-2: Volume characteristics

2.3 DEVICES

2.3.1 Which Devices are Available to User Jobs?

A user job is able to reserve for itself almost every peripheral device; exceptions may be found below.

Usually a user job can restrict itself to reserving tape and disk devices. Other devices such as printer and punch card devices are controlled by the system and may be addressed via the interfaces provided by the software product SPOOL (PRINT and PUNCH commands, WRLST macro, etc.).

Devices that **cannot** be reserved for individual jobs are:

- consoles
- disk devices for public volumes
- devices assigned to the system by the operator (e.g. printer and punch card devices).

These devices are system resources.

2.3.2 Requesting Devices

User jobs can request and reserve devices and volumes by means of the commands

FILE and
SECURE[-RESOURCE-ALLOCATION]

The SECURE[-RESOURCE-ALLOCATION] command can also be used to reserve other resources, e.g. files on public volumes.

If neither of these commands is used, the DMS attempts to reserve the required devices when opening a file.

Devices and volumes needed for a certain file are requested using the FILE command.

If a file which extends over more than one private volume is reserved by means of a SECURE[-RESOURCE-ALLOCATION] command, the DMS reserves all the volumes and the devices they require.

Furthermore, in batch mode, the SECURE[-RESOURCE-ALLOCATION] command causes the job to wait if one of the devices requested is not available at the time of the request. However, the maximum wait time can also be specified explicitly in the WAIT operand of the SECURE[-RESOURCE-ALLOCATION] command for both batch and interactive mode).

When requested, devices may be selected by means of the following attributes:

- According to the device type (applies to FILE and SECURE[-RESOURCE-ALLOCATION] commands; in this case, the DMS selects any free device of the specified type.

  Example:

  /FILE DATEI,LINK=EIN,DEVICE=D3465,VOLUME=DSK001
  /SEC-RES DISK=(VOL=DSK002,T=D3480)

  or

  /SECURE VOL=DSK002/D3480

  Using the mnemonic device name (applies only to the SECURE command). If it is free, the device with the designation specified is reserved by the DMS. However, this sort of reservation is possible for SPECIAL applications only.

  Note:

  The reservation of disk devices according to type by the SECURE[-RESOURCE-ALLOCATION] command is possible for SPECIAL applications only (e.g. VOLIN); for example: /SEC D3465=2).

  Example:

  /SECURE UNIT=A1

2.3.3 Reserving Devices Using the Data Management System

The DMS processes device requests as follows:

<u>If the requested devices are free</u>:

The DMS reserves the devices for the requesting job. If volumes are requested at the same time, the DMS sends a mount request to the operator (MOUNT message displayed on the console). For files on private disks, all disks containing parts of the relevant file must be mounted.

<u>If at least one of the devices requested is not free</u>:

In interactive mode an appropriate message is sent to the terminal. No devices are reserved. This applies regardless of whether a FILE command or a SECURE[-RESOURCE-ALLOCATION] command was used to request them.

If a SECURE[-RESOURCE-ALLOCATION] command was used for the request, the job is placed in the SECURE queue in accordance with the values in the WAIT operand and with its type (batch, dialog). Processing of the job continues when all the requested devices are free. If a FILE command was used or if not all the requested devices are free after the maximum wait time in a SECURE[-RESOURCE-ALLOCATION] command, all subsequent commands until the next STEP or LOGOFF command are skipped.

2.3.4 Releasing Devices

Devices assigned to a file may be deallocated by means of the RELEASE command.

2.3.5 Methods of Requesting Devices for Jobs in which Class 1 Programs are Executed

These jobs can also use the FILE and SECURE[-RESOURCE-ALLOCATION] commands to request devices.

In addition, they have the following options:

- Using the TOS FILE command to request assignment before program execution.
- Using the ASSGN macro to request assignment during program execution (see Appendix A.5).
- If the program employs the physical level of the FCP and does not use either of the above options to request devices, the required devices, if free, are assigned by the DMS when processing the first EXCP or EXCPW macro.

The DDEV (TOS) or CLOSE (TOS) macro can be used to release devices from a class 1 program.

2.3.6 Operator Intervention

If a job has successfully requested devices and volumes, intervention by the operator is necessary to mount volumes on the appropriate devices.

The DMS controls this activity by means of messages that request the operator to mount a volume on a particular device (MOUNT messages). Each of these messages contains the volume serial number (VSN) of the volume concerned and the mnemonic device name of the device already reserved by the DMS for the requesting job.

The operator can react as follows:

- by mounting the volume on the designated device;
- by mounting the volume on another device of the same type as specified in the message.

Any other operator reaction is rejected by the DMS, e.g. he does not mount the volume and thus causes an error message to be sent to the user.

2.3.7 Management of Private Disks

The user sees only the reservation of the volume, the required device being managed by the system. The device is not reserved.

All reservation functions executed as a result of DMS calls are interpreted as requests for shareable reservation.

The SECURE[-RESOURCE-ALLOCATION] command offers the possibility of reserving a private volume either exclusively or as shareable.

The control and monitoring of volumes can be set and modified by means of operator commands (cf. section 2.3.8).

Examples:

- mounting, dismounting, remounting of volumes
- setting of availability specifications independent of the online status
- control of exclusive/shareable operation of volumes (including SPDs)
- defining the time at which a device is to be reserved/released by a volume
- permanent or temporary removal of volumes
- consistency of disk locks

Device management offers online monitoring of the volumes in use.

2.3.8 Volume Monitoring

Volume monitoring supports and monitors disk utilization, i.e.

- the mounting of disks prior to use
- the remounting of devices during use
- the protection of disks against errored device access (data security and data protection).

To ensure the availability of the system function "volume monitoring", it has a permanent task for disk monitoring. Operations that temporarily affect availability are executed by temporary tasks.

3 COMMANDS

This chapter contains descriptions of the commands for data and device management.
All commands and operands can be truncated (right to left) as long as they remain unambiguous.

3.1 DATA MANAGEMENT

CATALOG PROCESS CATALOG ENTRY

The CATALOG command is used to create or update the catalog entry for a file or for a file generation group (FGG); in particular, it allows the user to specify file protection measures.
The user can convert a temporary file into a permanent one or vice versa.

Format 1 for the catalog entry of single files

Operation	Operands
{CATALOG | CAT}	pathname1[,filename2] [,STATE={N[EW] | U[PDATE]}][,ACCESS={WRITE | READ}][,SHARE={NO | YES}] [,WRPASS={NONE | password1}][,RDPASS={NONE | password2}] [,EXPASS={NONE | password3}][,RETPD=days][,BACKUP={A | B | C | D | E}] [,LARGE={YES | NO}][,DESTROY={YES | NO}][,AUDIT={SUCC | FAIL | ALL | NONE}]

"pathname1" stands for: [:catid:][$userid.]filename1

catid — This operand denotes the catalog identifier. If no entry is made, the default value assigned to the user ID is assumed.

userid — This operand denotes the user identification. If this operand is omitted, the user's own user ID is assumed. Only the user's own user ID is allowed.

filename1 — This operand specifies the fully qualified name of the file or temporary file to which the subsequent operands refer. The name of a file generation is allowed only for the purpose of creating its catalog entry (see the operand STATE=NEW).

filename2 — This operand specifies the new fully qualified name of a file or temporary file to which "filename1" is to be changed. "catid" and "userid" must not be specified. The command is rejected if STATE=UPDATE is not specified.

STATE=N[EW] — This operand specifies that a new catalog entry is to be created, i.e. the other operands in the CATALOG command describe a file which does not yet exist.

=U[PDATE] — This operand specifies that an existing catalog entry is to be modified. A field in the catalog entry can be modified only if the appropriate operand is specified explicitly, since omitted operands are not replaced by their default values.

If the file is password-protected, the password required for write access must be specified in order to update the catalog entry. The question of which passwords are to be specified for a specific access is discussed in section 2.1.5 (Table 2.1).

ACCESS=WRITE — This operand specifies that both read and write access are allowed for this file.

=READ — This operand specifies that only read access is permitted. This entry must be specified in conjunction with the operand STATE=UPDATE, i.e. it affects only catalog entries which already exist.

SHARE=NO — This operand restricts access to the file or its catalog entry to the owner only, i.e. the file is non-shareable.

=YES — This operand specifies that the file or its catalog entry may be processed under any user ID, i.e. the file is shareable.

WRPASS=NONE — No write password is required to write to the file.

=password1 — This operand defines the password required for writing to the file. Up to 4 bytes can be specified for the file protection passwords in the operands RDPASS, WRPASS and EXPASS:

C'x' "x" represents 1 to 4 alphanumeric characters and special characters;
X'n' "n" represents 1 to 8 hexadecimal digits;
d "d" represents 1 to 8 decimal digits whose value is converted to a binary value.

When a CATALOG command is output to the system file SYSLST (job logging) or SYSOUT (in the case of procedures), all passwords are overwritten with the letter "S".

Protection of a file by passwords is enhanced if the system uses password encryption. This means, for example, that only encrypted passwords are contained in memory dumps; misuse is thus made impossible.

At system generation, password encryption can be preset (system parameter ENCRYPT).

RDPASS=NONE — No read password is necessary to read the file.

=password2 — This operand defines the password required in order to read the file.

EXPASS=NONE — No password is necessary to execute the file (load module, DO or ENTER procedure).

=password3 — This operand defines the password required for execution of a file.

RETPD=days — This operand specifies the file retention period in days, during which time the file can be read only.
"days" must be a decimal integer, the maximum value of which is calculated from the difference in days between the end of the century (31.12.99) and the current date.
If STATE=NEW is specified, this operand is ignored.
If STATE=U is specified, this retention period can be specified for existing files only, i.e. the creation date must be greater than 0 (CRDATE field in the output of the FSTATUS command). Once the retention period has expired, the file can be modified, but it is not automatically deleted.
Default value: zero days, i.e. the file can be modified immediately.

BACKUP= — This operand specifies the frequency with which the file is to be saved (creation of backup). This information is needed for the regular save runs performed by the file saving system ARCHIVE.

=A — Most frequent saving. Files defined thus are saved in every save run.

=B — Files defined thus are saved if a save run for files with BACKUP=B or C or D takes place.

=C — Files defined thus are saved if a save run for files with BACKUP=C or D takes place.

=D — Least frequent saving. Files defined thus are saved only if a save run for files with BACKUP=D takes place.

=E — No saving by ARCHIVE, except in the case of DEXPORT. (This is relevant for work files, for example.)

Default value:
The default value for BACKUP is determined by the system administrator at system generation time.

LARGE=YES — In save runs using ARCHIVE, only those PAM blocks are saved which have been changed since the last full save operation. This is of particular relevance for large files.

=NO — The file is saved in its entirety in every save run using ARCHIVE.

DESTROY=YES — Whenever a file is deleted, its storage space is overwritten with binary zeros.

=NO — When the file is deleted, its storage space is returned unchanged to the system unless data destruction was explicitly requested (DESTROY operand in the ERASE command).

AUDIT= — This operand monitors file access operations by means of system exit routines for the specified file. CATALOG, ERASE, FILE and OPEN operations may be monitored.
This monitoring of file access operations can be restricted to specific user IDs by means of the operand AUDIT in the JOIN command (see the "System Exits" manual.

=SUCC — All successful DMS actions for the file are to be monitored.

=FAIL — All unsuccessful DMS actions for the file are to be monitored.

=ALL — All DMS actions for the file are to be monitored.

=NONE — No monitoring.

Notes on temporary files:

- The RETPD operand has no effect.
- The following system default values apply and cannot be modified:

  RETPD=0, RDPASS=NONE, WRPASS=NONE, SHARE=NO, ACCESS=WRITE, BACKUP=E

- When recataloging a temporary file as a permanent file: the file attributes are taken over.
  When recataloging a permanent file as a temporary file: the default values are used for the attributes.

Format 2 for the catalog entry of file generation groups

Operation	Operands
{CATALOG / CAT}	pathname1[,filename2]
	[,STATE={N[EW] / U[PDATE] / F[OREIGN]}]
	[,ACCESS={WRITE / READ}]
	[,SHARE={NO / YES}]
	[,WRPASS={NONE / password1}]
	[,RDPASS={NONE / password2}]
	[,RETPD=days][,GEN=max]
	[,DISP={CYCLE / REUSE / DELETE / KEEP}]
	[,BASE={absbas / relbas}]
	[,FIRST=n][,DEVICE=device][,VOLUME=vsn]
	[,BACKUP={A / B / C / D / E}]
	[,LARGE={YES / NO}]
	[,DESTROY={YES / NO}]
	[,AUDIT={SUCC / FAIL / ALL / NONE}]

"pathname1" stands for: [:catid:][$userid.]filename1

catid — This operand denotes the catalog identifier. If no entry is made, the default catalog ID assigned to the user ID is assumed.

userid — This operand denotes the user identification. If this operand is omitted, the user's own user ID is assumed. Only the user's own user ID is allowed.

filename1 — This operand specifies the name of a file generation group to which the subsequent operands refer.

filename2 — This operand specifies a new name for file generation group "filename1". The user ID and/or the catalog identifier must not be specified.

The command is rejected if STATE=UPDATE is not specified.

STATE=N[EW] — This operand specifies that a new catalog entry is to be created, i.e. the other operands in the CATALOG command describe a file generation group which does not yet exist.

=U[PDATE] — This operand specifies that an existing catalog entry is to be updated. If no entry is made, the default value is not assumed. Thus, a field in the catalog can be modified only if the appropriate operand is specified explicitly.

If the file is password-protected, the password required for write access must be specified in order to update the catalog entry. The question of which passwords are to be specified for a particular access is described in section 2.1.5 (Table 2-1).

=F[OREIGN] — This operand must be specified for a file generation group if the latter is on private disks and does not yet have a catalog entry (FOREIGN file generation group). In addition to this operand, the DEVICE and VOLUME operands must be specified in the CATALOG command.

In addition, a FILE command with the entries "STATE=FOREIGN,DEVICE=...,VOLUME=..." must be given for each individual generation within a FOREIGN file generation group.

The following table illustrates the permissible combinations of STATE=NEW, UPDATE or FOREIGN with the other operands in the CATALOG command:

Operands	NEW	STATE=UPDATE	FOREIGN
ACCESS	-	X	-
SHARE	X	X	-
passwords	X	X	-
RETPD	-	X	-
GEN	X	X	-
DISP	X	X	-
BASE={absbas / relbas}	X	X	-
FIRST	X	-	-
DEVICE	X	-	X
VOLUME	X	-	X

ACCESS=WRITE — This operand specifies that both read and write access are allowed for this file generation group.

=READ — This operand specifies that only read access is permitted. This entry must be specified in conjunction with the operand STATE=UPDATE, i.e. it can be specified only where the catalog entry already exists.

SHARE=NO — This operand restricts access to the file generation group or its catalog entry to the owner, i.e. the file is non-shareable.

=YES — This operand specifies that the file generation group or its catalog entry may be processed under any user ID, i.e. the file is shareable.

WRPASS=NONE — No write password is required to write to the generations of the file generation group.

=password1 — This operand defines the password required for writing (see format 1)

RDPASS=NONE — No read password is necessary to read the generations of the file generation group.

=password2 — This operand defines the password required for reading.

RETPD=days — This operand specifies the file retention period in days, during which time the file can be read only.
"days" must be a decimal integer, the maximum value of which is calculated from the difference in days between the end of the century (31.12.99) and the current date.
If STATE=NEW is specified, this operand is ignored.
If STATE=U is specified, this operand is effective for generations whose creation date is greater than 0, but is ignored for all other generations.
Once the retention period has expired, the file can be updated, but it is not automatically deleted.
Default value: zero days, i.e. the file can be updated immediately.

GEN=max

This operand specifies the maximum number of generations which can be cataloged simultaneously for the file generation group "filename1". The maximum value is 255 and should not be equal to 0.

If STATE=NEW and GEN=0 are specified, a file is created instead of a file generation group.
The GEN operand can be specified either for a new file generation group (STATE=NEW) or in conjunction with STATE=UPDATE.

DISP=

This operand specifies what is to happen when the number of cataloged file generations reaches the maximum value defined in GEN and a further file generation is to be cataloged.

=CYCLE

This operand specifies that the oldest generation (see the FIRST operand) and its catalog entry are to be deleted.

=REUSE

The same as for CYCLE, provided that the generations of the file generation group are located on **public volumes.**
If the generations of the file generation group are located on **private disks,** the oldest generation is deleted and the new one is created on the same volume. If the deleted generation extended over a number of disks, only the first disk is used to catalog the new generation.

=DELETE

This operand specifies that all existing file generations are to be deleted. A new series of generations is thus started as soon as the maximum value specified in GEN has been exceeded.

=KEEP

This entry suppresses the automatic deletion of generations older than that most recently processed, even if the maximum number of generations is exceeded in doing so. The FIRST and BASE values can be modified only by means of the CATALOG command with STATE=U. This command also effects the deletion of the surplus older generations.

BASE=

This operand specifies the base value for file generation group "filename1", to which all relative generation numbers within a job refer.

If the BASE operand is omitted, it receives the following value by default:

- value of the FIRST operand if no generation is present;
- zero, if no generation is present and no FIRST operand was specified.

=absbas

This operand specifies a decimal number (1 to 4 digits) greater than 0, which is to become the new base value for the file generation group.

If "absbas" is specified in conjunction with STATE=UPDATE, the new base value, which must define an existing file generation, is inserted in the catalog entry of the file generation group.

If BASE=absbas is specified in conjunction with STATE=NEW, absbas is inserted as the base value in the catalog entry and as the latest generation number (LASTGN field in the FSTAT output). A CAT command with the BASE operand must be specified if several file generations are to be imported, which are contained on private disk and whose group entry no longer exists (in the catalog and in the F1 label). For an example see the IMPORT command.

=relbas

This operand specifies a new base value for the file generation group. "relbas" must be ≤ 0 and must not exceed two digits. BASE is derived as follows:
BASE = (latest generation number + "relbas")

BASE=relbas can be specified only in conjunction with STATE=UPDATE. The new base value must define an existing file generation, i.e. positive numbers must not be used.

FIRST=n

This operand specifies the number of the oldest existing file generation, where "n" is a decimal number of 1 to 4 digits. The value of "n" must be less than that in the operand BASE, and the difference between these values must not exceed the number specified by GEN.
The operand FIRST can be specified only for file generation groups to be cataloged for the first time, i.e. in conjunction with STATE=NEW.

Note:
The operand FIRST is required in order to restore the catalog entry of a FOREIGN file generation group on private volumes. It should be used for this purpose only.

DEVICE=device

This operand specifies a disk device. For a formal description, see the DEVICE operand in the FILE command.

VOLUME=vsn

This operand specifies the volume serial number of the private disk on which the catalog entry of the file generation group is located, or is to be located.

The operands VOLUME and DEVICE must be specified in conjunction:

- with STATE=NEW in order to create a file generation group on private disks;
- with STATE=FOREIGN to restore a file generation group which already exists on private disks. (In this case, the BASE, FIRST, GEN and DISP entries are transferred from the F1 label to the catalog.)

If the file generation group is cataloged on a private disk, all associated generations must also be located on private disks, and be generated by an appropriate FILE command or FILE macro.

BACKUP= This operand specifies the frequency with which the file is to be saved (creation of backup). This information is needed for the regular save runs performed by the file saving system ARCHIVE.

=A Most frequent saving; files thus defined are saved in every save run.

=B Files defined thus are saved if a save run for files with BACKUP=B or C or D takes place.

=C Files defined thus are saved if a save run for files with BACKUP=C or D takes place.

=D Least frequent saving; files defined thus are saved only if a save run for files with BACKUP=D takes place.

=E No saving by ARCHIVE. (This is of relevance for work files, for example.)

Default value:
The default value for BACKUP is defined by the system administrator at system generation time.

LARGE=YES In save runs using ARCHIVE, only those PAM blocks which have been modified since the last save operation are saved. This is of particular relevance for large files.

=NO The file is saved in its entirety in every save run using ARCHIVE.

DESTROY=YES Whenever the file is deleted, its storage space is overwritten with binary zeros.

=NO When the file is deleted, its storage space is returned unchanged to the system unless data destruction was specifically requested (DESTROY operand in the ERASE command).

Note for the BACKUP, LARGE and DESTROY operands:
The values of these operands apply to the file generation group as a whole. It is not possible to specify these attributes separately for individual generations.

AUDIT= This operand monitors the file access operations by means of system exit routines for the named file.
CATALOG, ERASE, FILE and OPEN operations can be monitored.

This monitoring of file access operations can be restricted to specific user IDs by the AUDIT operand in the JOIN command (see the "System Exits" manual).

=SUCC All successful DMS actions for the file are to be monitored.

=FAIL All unsuccessful DMS actions for the file are to be monitored.

=ALL All DMS actions for the file are to be monitored.

=NONE No monitoring.

Notes:

- Only the attributes of the group can be modified for file generations of a file generation group. Equally, the group attributes must already be defined before file generations are added to this group (see the FOREIGN entry in the FILE command).

- When the output generated by a CATALOG command is directed to the system file SYSLST (job logging) or SYSOUT (in the case of procedures), all passwords are overwritten with the letter "S".

- A file can be recataloged as a file generation; it need not be copied. This applies only if the file generation does not yet exist. However, a file generation cannot be recataloged as a file.

Network catalog management (see also the "MSCF Multiprocessor System" manual)

The user of the multiprocessor system can specify a unique "path name", which also contains a catalog identifier, as a supplement to the currently existing file name.

The catalog identifier and the user ID must not be modified when changing this name, i.e. it is not possible to specify a catalog identifier and/or a user ID in "filename2".

Example 1: Format 1

```
/FSTAT CAT.TEST
%  DMS0533 REQUESTED FILE NOT CATALOGED ON PVS V. CMD TERMINATED
/CAT CAT.TEST ———————————————————————————————————————————————— 1)
/FSTAT CAT.TEST,ALL
0000000 :V:$PM211034.CAT.TEST
  FCBTYPE = NONE     VSNTYPE = PUB
  SHARE   = NO       ACCESS  = WRITE
  ACCESS# = 000      CRDATE  = NONE      EXDATE  = NONE      LADATE  = NONE
  RDPASS  = NONE     WRPASS  = NONE      EXPASS  = NONE
  VERSION = 000      BACKUP# = 000       LARGE   = NO        BACKUP  = A
  DESTROY = NO       AUDIT   = NONE
  BLKTYPE = NONE     BLKSIZE = 000000    RECFORM = NONE      RECSIZE = 00000
  VSN/DEV/EXT =      NONE
:V:  PUBLIC:       1 FILE.  RES=        0, FREE=        0, REL=        0 PAGES
```

1) The CATALOG command catalogs the file CAT.TEST and enters the default values, as shown in the subsequent FTATUS command.

Example 2: Format 1 of the CATALOG command

The three files PROG1, PROG2 and PROG3 contain executable programs:

```
/CAT PROG1,RDPASS=NONE,EXPASS=NONE,STATE=UPDATE ———————————— 1)
/CAT PROG2,RDPASS=X'FF11',EXPASS=NONE,STATE=U ———————————— 2)
/CAT PROG3,RDPASS=C'LIES',EXPASS=X'123456',STATE=U ———————— 3)
```

1) File PROG1 can be read and executed.

2) File PROG2 cannot be read without knowledge of the read password X'FF11', but the load module in file PROG2 can be loaded with the EXEC or LOAD command and then started.

3) The system permits processing of file PROG3 only if it has knowledge of at least one of the passwords C'LIES' and X'123456'. If a user knows only password X'123456', he can execute the program on the file but he cannot read the file. If a user knows the read password C'LIES', this suffices to allow him to read and execute the file.

Example 3: Format 1 of the CATALOG command

The file DO.BEISPIEL contains a DO procedure.

```
/DO DO.BEISPIEL                                                    1)
%  BLS0517 MODULE TEST LOADED.
PROZEDUR LAEUFT                                                    2)
/CAT DO.BEISPIEL,EXPASS=C'NOEX',STATE=U                            3)
/DO DO.BEISPIEL
%  EXC0141 PROCEDURE FILE DO.BEISPIEL CAN'T BE OPENED. DMS ERROR CODE 0D91
   CMD TERM
/PASSWORD C'NOEX'                                                  4)
/DO DO.BEISPIEL
%  BLS0517 MODULE TEST LOADED.
PROZEDUR LAEUFT
```

1) Since the file DO.BEISPIEL is not protected by an execute password, it may be executed without specification of a password.

2) Procedure executing.

3) The file is given an execute password. From now on it cannot be executed unless this password is specified.

4) After the password has been entered, the file can be executed again.

Example 4: Format 2 of the CATALOG command

A file generation group (FGG or DGG) was defined as follows:

/CAT GROUP,FIRST=1,GEN=3,BASE=3,DISP=KEEP

The command sequence below then works out as follows:

Command:	BASE:	FIRST:	LAST:	Absolute number
/LOGON ...	3	1	3	0
/FILE GROUP(+1)	3	1	4	4
/FILE GROUP(-1)	3	1	4	2
/FILE GROUP(+2)	3	1	5	5
/EXEC ...				
/CAT GROUP,STATE=U, BASE=0	5	3	5	5

The older generations, 1 and 2, are deleted by the CATALOG command; the new BASE value is 5; the new FIRST value is derived from the following:

BASE(5) - GEN(3) + 1 = FIRST(3) ;

Example 5: Format 2

```
/CAT DGG1,GEN=5 ———————————————————————————————————————————————————— 1)
/FSTAT DGG1,ALL
0000000 :V:$PM211034.DGG1 (FGG)
  SHARE    = NO       ACCESS   = WRITE
  ACCESS#  = 000      CRDATE   = 85-03-19  EXDATE   = 85-03-19  LADATE   = NONE
  RDPASS   = NONE     WRPASS   = NONE      EXPASS   = NONE
  VERSION  = 000      BACKUP#  = 000       LARGE    = NO        BACKUP   = A
  DESTROY  = NO       AUDIT    = NONE
  GEN      = 00005    BASE     = 00000     LASTGN   = 00000     FIRSTGN  = 00000
  DISP     = CYCLE
:V:  PUBLIC:       1 FILE.  RES=        0, FREE=        0, REL=        0 PAGES
/CAT DGG1(*1)
/CAT DGG1(*3) ———————————————————————————————————————————————————— 2)
%  DMS06C7 USER ATTEMPTED TO PROCESS A GENERATION WITH AN INCORRECT GENERATION
NUMBER. COMMAND TERMINATED
/CAT DGG1(*2)
/CAT DGG1(*3)
/CAT DGG1(*4)
/CAT DGG1(*5)
/FSTAT DGG1,ALL
0000000 :V:$PM211034.DGG1 (FGG)
  SHARE    = NO       ACCESS   = WRITE
  ACCESS#  = 000      CRDATE   = 85-03-19  EXDATE   = 85-03-19  LADATE   = NONE
  RDPASS   = NONE     WRPASS   = NONE      EXPASS   = NONE
  VERSION  = 000      BACKUP#  = 000       LARGE    = NO        BACKUP   = A
  DESTROY  = NO       AUDIT    = NONE
  GEN      = 00005    BASE     = 00000     LASTGN   = 00005     FIRSTGN  = 00001
  DISP     = CYCLE
:V:  PUBLIC:       1 FILE.  RES=        0, FREE=        0, REL=        0 PAGES
/CAT DGG1(*6) ———————————————————————————————————————————————————— 3)
/FSTAT DGG1,GEN=YES
0000000 :V:$PM211034.DGG1  (FGG)
0000000 :V:$PM211034.DGG1(*0002)
0000000 :V:$PM211034.DGG1(*0003)
0000000 :V:$PM211034.DGG1(*0004)
0000000 :V:$PM211034.DGG1(*0005)
0000000 :V:$PM211034.DGG1(*0006)
:V:  PUBLIC:       6 FILES. RES=        0, FREE=        0, REL=        0 PAGES
/FSTAT DGG1,ALL
0000000 :V:$PM211034.DGG1 (FGG)
  SHARE    = NO       ACCESS   = WRITE
  ACCESS#  = 000      CRDATE   = 85-03-19  EXDATE   = 85-03-19  LADATE   = NONE
  RDPASS   = NONE     WRPASS   = NONE      EXPASS   = NONE
  VERSION  = 000      BACKUP#  = 000       LARGE    = NO        BACKUP   = A
  DESTROY  = NO       AUDIT    = NONE
  GEN      = 00005    BASE     = 00000     LASTGN   = 00006     FIRSTGN  = 00002
  DISP     = CYCLE
:V:  PUBLIC:       1 FILE.  RES=        0, FREE=        0, REL=        0 PAGES
```

1) A maximum of 5 file generations are to be cataloged for the file generation group DGG1.

2) The file generations must be cataloged in unbroken ascending order.

3) If the number of file generations exceeds the value laid down in the GEN operand, the oldest generation is deleted in each case (here (*1)).

For further examples see the PASSWORD command.

CHANGE CHANGE TFT ENTRY

The CHANGE command changes the file link name in a Task File Table (TFT) entry. All other values in the entry remain unaltered.

Operation	Operands
CHANGE	[link1],link2

link1 This operand specifies the file link name (1 to 8 bytes).

If this operand is omitted, the first TFT entry with the link name C'␣␣␣␣␣␣␣␣' is processed.

link2 This operand specifies a new file link name which is to replace the previous file link name.

Note:

The CHANGE command is meaningful only for a TFT entry which was previously created by means of a FILE command/macro (and not with OPEN).
The reason for this is the differing treatment of the TFT entry upon reopening the file (see section 5.1, "Opening Files").

Example:

The module UNSORT generates an output file, which is subsequently to be sorted ("SORTIERT") by means of the SORT utility routine.

```
/FILE DATEN.UNSORTIERT,LINK=UNSORT,SPACE=12 ———————————————— 1)
/EXEC (UNSORT,RNO.MODLIB)
%  BLS0001 DLL VER 822
/CHANGE UNSORT,SORTIN ———————————————————————————————— 2)
/FILE DATEN.SORTIERT,LINK=SORTOUT
/EXEC $SORT
%  BLS0500 PROGRAM SORT, VERSION 710 OF 84-12-17 LOADED.
%  SRT1001 12:55:04/000000.00 SORT/MERGE STARTED, VERSION 71A00
%  SRT1130 PLEASE ENTER SORT STATEMENTS
*sort fields=(1,4,a,ch)
*end
%  SRT1116 NUMBER OF OUTPUT RECORDS - 00000005000
%  SRT1116 NUMBER OF INPUT  RECORDS - 00000005000
%  SRT1116 NUMBER OF SORTED RECORDS - 00000005000
%  SRT1104 CPU TIME USED: 0002.7
/REL SORTOUT
```

1) This FILE command links the file DATEN.UNSORTIERT (=data.unsorted) with the module UNSORT; this module uses the link name UNSORT.

2) The CHANGE command assigns the file DATEN.UNSORTIERT to the software product SORT as an input file, i.e. the file link name UNSORT is replaced by SORTIN.

For a further example see the HOLD command.

COPY COPY FILE

The COPY command copies files, file generations or file generation groups, without altering them, from:

- disk to disk
- disk to tape
- tape to disk.

In order for the COPY command to be able to process them, tape files must have the standard block format (multiples of 2048 bytes).

Operation	Operands
COPY	pathname1,pathname2[,SAME] [,WRITE={REPL[ACE] | NEW[,DIA[LOG]={Y[ES] | N[O]}]}]

"pathname1" stands for: [:catid:][$userid.]filename1
"pathname2" stands for: [:catid:][$userid.]filename2

catid — This operand denotes the catalog identifier associated with the file. If no entry is made, the default catalog ID assigned to the user ID is assumed.

userid — This operand denotes the user identification associated with the file. If this operand is not specified, the user's own user ID is assumed.

filename1 — This operand specifies the fully qualified name of a file, file generation or file generation group which is to be copied.
If the file does not belong to the user's own user ID, it must be shareable (SHARE=YES) and the user ID must be specified.

filename2 — This operand specifies the name which the copy (file, file generation or file generation group) is to bear.

If "filename2" has not yet been cataloged, the file name must not start with a foreign user ID, and is to be entered without it (cf. example 2). If the file does not belong to the user's own user ID, it must be declared shareable (operand SHARE=YES in the CAT command) and the user ID must be specified.

If "filename2" is the name of a file generation group, this must also be true for "filename2", i.e. a file or file generation cannot be copied to a file generation group.

Note:

"pathname1" must not be identical with "pathname2".

SAME

This operand specifies that "filename2" is to receive the same file protection attributes as "filename1" (same passwords, same retention period, same shareability, same access authorization).
If the SAME entry is omitted, a file ("filename2") to be cataloged is created as a non-shareable file (SHARE=NO) and is free for both read and write access.
SAME is ignored if the output file is a file generation, since its file protection is determined by the associated file generation group. It is also ignored if the output file is cataloged under another user's user ID.

Note:
The value for AUDIT from the catalog entry, which organizes the file accesses via exit routines, is not transferred to the copy with SAME.

WRITE=

If the file, file generation or file generation group defined by "pathname2" already exists and is not empty, the user can determine whether or not this file is overwritten.

=REPL[ACE]

The file, file generation or file generation group defined by "pathname2" is overwritten without a message (default value).

=NEW

The file is not copied; the command is rejected. In the case of procedures, the message "pathname2 ALREADY EXISTS. COMMAND TERMINATED" is displayed and a branch is made to the next STEP command.

DIA[LOG]=

This operand determines whether or not the query "OVERWRITE (Y/N)?" is displayed, and is interpreted only in interactive mode and if WRITE=NEW is specified.

=Y[ES]

The user is asked if the file is to be overwritten.

=N[O]

The existing file is not overwritten. The message "pathname2 ALREADY EXISTS. COMMAND TERMINATED" is displayed.

Notes:

- The COPY command copies files in blocks (1 PAM block = 2048 bytes) and cannot therefore be used to modify file attributes during copying. It cannot, for instance, convert a SAM file into an ISAM file, or fixed-length records into variable-length records.

- The COPY command is rejected if output file "filename2" is free for read access only (ACCESS=READ in the CATALOG command), or if the secondary allocation of the output file is 0 and the primary allocation is inadequate.

  If the file "filename2" has not yet been cataloged, this is effected during processing of the COPY command. This output file is subsequently created on public disks under the user's own user ID.

  If output file "filename2" is to be placed on private disk and has not yet been cataloged, the user must issue an appropriate FILE command for the output file prior to the COPY command.

- If the file to which the copy is to be made has not yet been cataloged, the secondary allocation is taken over from the original disk file. If the original file is on tape, default values are assumed for primary and secondary allocation.

- Only a file generation group which consists of SAM file generations with the same file characteristics (same record and block length and the same record format) can be copied to a single file or file generation. In this case, the file generation must not belong to the file generation group which is to be copied.

- A file generation group can be copied into another file generation group only if one of the following conditions is fulfilled:

  a) The group entries for both file generation groups match (i.e. the values of GEN, FIRST, LASTGN and BASE are the same).
     The file generation group into which the DMS enters the copy must already contain all the generations from FIRST to LASTGN (i.e. the generations must be cataloged and be provided with storage space).

  b) The value of GEN is the same for both file generation groups, and the file generation group into which the DMS writes the copy does not at this stage contain any other generations (i.e. FIRST, LASTGN, and BASE have the value zero).

- If a file is to be copied from a foreign user ID "userid", the user ID "$userid" must precede "filename1". The file must be shareable (SHARE=YES in the catalog entry).

- If a file on private disk has an entry in the system catalog but not in the F1 label, this catalog entry is deleted. If the file is the input file, the command is rejected. If it is an output file, a new file is created in public space.

- If the file/generation defined by "pathname2" is a tape file, the WRITE operand is ignored, i.e. the file/generation is copied whatever happens.

Remote file access (see also the "RFA" manual)

- If a file is to be copied from one remote system to another, with input/output being on 2 different systems, this is supported by the higher-ranking execution routine. The local system serves as an intermediate station for the data transfer.

- Copying is possible only if the RFASTART command is issued for both remote systems before the copying process commences.

- Copying of a local file generation group from a remote system is not possible (error message D5FF).

- When copying a remote file to a local file with the SAME operand, the passwords are not taken over.

Example 1:

The file DATEI is copied to the file DATEI.KOPIE.

```
/FSTAT DATEI,C
0000006 :V:$PM211034.DATEI
  SHARE   = YES      ACCESS  = WRITE
  ACCESS# = 001      CRDATE  = 85-03-19  EXDATE  = 85-03-19  LADATE = 85-03-19
  RDPASS  = YES      WRPASS  = YES       EXPASS  = YES
  VERSION = 001      BACKUP# = 000       LARGE   = NO        BACKUP = A
  DESTROY = NO       AUDIT   = NONE
:V:  PUBLIC:      1 FILE.  RES=        6, FREE=        2, REL=       0 PAGES
/PASSWORD C'DAT'
/COPY DATEI,DATEI.KOPIE ------------------------------------------------- 1)
/FSTAT DATEI.KOPIE,C
0000006 :V:$PM211034.DATEI.KOPIE
  SHARE   = NO       ACCESS  = WRITE
  ACCESS# = 001      CRDATE  = 85-03-19  EXDATE  = 85-03-19  LADATE = 85-03-19
  RDPASS  = NONE     WRPASS  = NONE      EXPASS  = NONE
  VERSION = 001      BACKUP# = 000       LARGE   = NO        BACKUP = A
  DESTROY = NO       AUDIT   = NONE
:V:  PUBLIC:      1 FILE.  RES=        6, FREE=        2, REL=       0 PAGES
/COPY DATEI,DATEI.KOPIE,SAME ------------------------------------------- 2)
/FSTAT DATEI.KOPIE,C
0000006 :V:$PM211034.DATEI.KOPIE
  SHARE   = YES      ACCESS  = WRITE
  ACCESS# = 002      CRDATE  = 85-03-19  EXDATE  = 85-03-19  LADATE = 85-03-19
  RDPASS  = YES      WRPASS  = YES       EXPASS  = YES
  VERSION = 001      BACKUP# = 000       LARGE   = NO        BACKUP = A
  DESTROY = NO       AUDIT   = NONE
:V:  PUBLIC:      1 FILE.  RES=        6, FREE=        2, REL=       0 PAGES
```

1) The COPY command generates the file DATEI.KOPIE as a copy of the file DATEI. The protection attributes of the file are, however, not adopted.

2) As a result of the SAME entry, the file DATEI.KOPIE receives the same protection attributes as DATEI: SHARE, WRPASS, RDPASS, EXPASS.

Example 2:

Copying to/from a different user ID

```
/FSTAT $PM211023.
0000003 :V:$PM211023.ASS.BEISPIEL2
0000846 :V:$PM211023.DVS.BAND
0000015 :V:$PM211023.DVS.BEISP.3
0000006 :V:$PM211023.DVS.BEISP.5
0000006 :V:$PM211023.M.NEU
:V:  PUBLIC:      5 FILES. RES=      876, FREE=        6, REL=        3 PAGES
/COPY $PM211023.M.NEU,DATEN ———————————————————————————————————— 1)
/FSTAT DATEN
0000006 :V:$PM211034.DATEN
:V:  PUBLIC:      1 FILE.  RES=        6, FREE=        1, REL=        0 PAGES
/COPY DAT,$PM211023.ASS.BEISPIEL2 ———————————————————————————— 2)
/FSTAT $PM211023.
0000030 :V:$PM211023.ASS.BEISPIEL2
0000846 :V:$PM211023.DVS.BAND
0000015 :V:$PM211023.DVS.BEISP.3
0000006 :V:$PM211023.DVS.BEISP.5
0000006 :V:$PM211023.M.NEU
:V:  PUBLIC:      5 FILES. RES=      903, FREE=        8, REL=        3 PAGES
```

1) The file M.NEU with the user ID $PM211023 is copied to the file DATEI.

2) The file DAT is copied to the file ASS.BEISPIEL2 with the user ID $PM211023. The file ASS.BEISPIEL2 was previously cataloged with SHARE=YES.

Example 3:

File generation groups (FGG or DGG) and file generations

```
/FSTAT DGG1,ALL
0000000 :V:$PM211023.DGG1 (FGG)
  SHARE    = NO       ACCESS   = WRITE
  ACCESS#  = 000      CRDATE   = 85-03-20  EXDATE   = 85-03-20  LADATE   = NONE
  RDPASS   = NONE     WRPASS   = NONE      EXPASS   = NONE
  VERSION  = 000      BACKUP#  = 000       LARGE    = NO        BACKUP   = A
  DESTROY  = NO       AUDIT    = NONE
  GEN      = 00005    BASE     = 00000     LASTGN   = 00005     FIRSTGN  = 00001
  DISP     = CYCLE
:V:  PUBLIC:       1 FILE.  RES=        0, FREE=        0, REL=        0 PAGES
/CAT DGG2,GEN=5
/COPY DGG1,DGG2
/FSTAT DGG2,ALL
0000000 :V:$PM211023.DGG2 (FGG)
  SHARE    = NO       ACCESS   = WRITE
  ACCESS#  = 000      CRDATE   = 85-03-20  EXDATE   = 85-03-20  LADATE   = NONE
  RDPASS   = NONE     WRPASS   = NONE      EXPASS   = NONE
  VERSION  = 000      BACKUP#  = 000       LARGE    = NO        BACKUP   = A
  DESTROY  = NO       AUDIT    = NONE
  GEN      = 00005    BASE     = 00000     LASTGN   = 00005     FIRSTGN  = 00001
  DISP     = CYCLE
:V:  PUBLIC:       1 FILE.  RES=        0, FREE=        0, REL=        0 PAGES
```

The file generation group DGG1 with GEN=5 is copied to DGG2. Prior to this, DGG2 must be cataloged with the same GEN value (GEN=5).

```
/FSTAT DGG2,ALL
0000000 :V:$PM211023.DGG2 (FGG)
  SHARE    = NO       ACCESS   = WRITE
  ACCESS#  = 000      CRDATE   = 85-03-20  EXDATE   = 85-03-20  LADATE   = NONE
  RDPASS   = NONE     WRPASS   = NONE      EXPASS   = NONE
  VERSION  = 000      BACKUP#  = 000       LARGE    = NO        BACKUP   = A
  DESTROY  = NO       AUDIT    = NONE
  GEN      = 00005    BASE     = 00000     LASTGN   = 00005     FIRSTGN  = 00001
  DISP     = CYCLE
:V:  PUBLIC:       1 FILE.  RES=        0, FREE=        0, REL=        0 PAGES
/COPY DGG2,DGG3
%  DMS06D8 GENERATIONS DO NOT HAVE SAME ATTRIBUTES. COMMAND TERMINATED
/CAT DGG3,GEN=5
/COPY DGG2,DGG3
/FSTAT DGG3,ALL
0000000 :V:$PM211023.DGG3 (FGG)
  SHARE    = NO       ACCESS   = WRITE
  ACCESS#  = 000      CRDATE   = 85-03-20  EXDATE   = 85-03-20  LADATE   = NONE
  RDPASS   = NONE     WRPASS   = NONE      EXPASS   = NONE
  VERSION  = 000      BACKUP#  = 000       LARGE    = NO        BACKUP   = A
  DESTROY  = NO       AUDIT    = NONE
  GEN      = 00005    BASE     = 00000     LASTGN   = 00005     FIRSTGN  = 00001
  DISP     = CYCLE
:V:  PUBLIC:       1 FILE.  RES=        0, FREE=        0, REL=        0 PAGES
```

The file generation group DGG2 already exists and contains generations. In order for DGG2 to be copied to DGG3, the group entries of both file generation groups must tally.

```
/FSTAT DGG1,GEN=YES
0000000 :V:$PM211034.DGG1 (FGG)
0000003 :V:$PM211034.DGG1(*0001)
0000003 :V:$PM211034.DGG1(*0002)
0000003 :V:$PM211034.DGG1(*0003)
0000003 :V:$PM211034.DGG1(*0004)
0000003 :V:$PM211034.DGG1(*0005)
:V:  PUBLIC:      6 FILES. RES=      15, FREE=      10, REL=      0 PAGES
/COPY DGG1,DATEI1
/FSTAT DATEI1,ALL
0000006 :V:$PM211034.DATEI1
  FCBTYPE = SAM      VSNTYPE = PUB       LASTPG  = 0000005   2ND ALLO= 00009
  SHARE   = NO       ACCESS  = WRITE
  ACCESS# = 001      CRDATE  = 85-03-21  EXDATE  = 85-03-21  LADATE  = 85-03-21
  RDPASS  = NONE     WRPASS  = NONE      EXPASS  = NONE
  VERSION = 001      BACKUP# = 000       LARGE   = NO        BACKUP  = A
  DESTROY = NO       AUDIT   = NONE
  BLKTYPE = STD      BLKSIZE = 002048    RECFORM = (V,N)     RECSIZE = 00000
  VSN/DEV/EXT =      PUBV11/D3475/002
  EXTCNT  =   2
:V:  PUBLIC:      1 FILE.  RES=       6, FREE=       1, REL=      0 PAGES
```

The file generation group DGG1 contains 5 SAM file generations with matching file attributes (record and block length, record format etc.). They are copied together to the file DATEI1 by means of the COPY command.

```
/COPY DGG1(*2),DGG2(*6)
/FSTAT DGG2,GEN=YES
0000000 :V:$PM211023.DGG2 (FGG)
0000009 :V:$PM211023.DGG2(*0002)
0000012 :V:$PM211023.DGG2(*0003)
0000012 :V:$PM211023.DGG2(*0004)
0000006 :V:$PM211023.DGG2(*0005)
0000009 :V:$PM211023.DGG2(*0006)
:V:  PUBLIC:      6 FILES. RES=      48, FREE=       3, REL=      0 PAGES
```

The COPY command generates the file generation DGG2(*6) as a copy of file generation DGG1(*2).

Example 4:

```
/COPY DATEI,DATEI.KOPIE,WRITE=NEW  ———————————————————————————— 1)
%  DMS0518 FILE :V:$PM211034.DATEI.KOPIE ALREADY EXISTS. OVERWRITE? REPLY (YES=Y; NO=N)?
Y
/COPY DATEI,DATEI.KOPIE,WRITE=NEW  ———————————————————————————— 2)
%  DMS0518 FILE :V:$PM211034.DATEI.KOPIE ALREADY EXISTS. OVERWRITE? REPLY (YES=Y; NO=N)?
N
%  DMS0519 COPY COMMAND WITHDRAWN BY CALLER
/COPY DATEI,DATEI.KOPIE,WRITE=NEW,DIALOG=NO  ———————————————— 3)
%  DMS051A FILE :V:$PM211034.DATEI.KOPIE ALREADY EXISTS. COMMAND TERMINATED
/COPY DATEI,DATEI.KOPIE,WRITE=NEW,DIALOG=YES
%  DMS0518 FILE :V:$PM211034.DATEI.KOPIE ALREADY EXISTS. OVERWRITE? REPLY (YES=Y; NO=N)?
N
%  DMS0519 COPY COMMAND WITHDRAWN BY CALLER
/COPY DATEI,DATEI.KOPIE,WRITE=REPLACE,DIALOG=YES
/COPY DATEI,DATEI.KOPIE,WRITE=REPLACE,DIALOG=NO
/COPY DATEI,DATEI.KOPIE
```

1), 2) Message DMS0518 is displayed.
3) Message DMS0518 is suppressed. Message DMS051A is displayed.

DROP CANCEL HOLD STATUS FOR TFT ENTRY

The DROP command cancels the HOLD status imposed by the HOLD command on an entry in the Task File Table (TFT). If a RELEASE command (or REL macro) was pending for this entry, it is now processed, i.e. the TFT entry is deleted in accordance with the specification in the RELEASE command or REL macro, and the private devices associated with it are released.

Operation	Operands
DROP	[link]

link

This operand specifies the file link name, i.e. the name of the Task File Table entry for which the HOLD status has been canceled.
If this entry is omitted, the first TFT entry with the LINK name C'␣␣␣␣␣␣␣␣' is processed.

For examples see the HOLD command.

ERASE ERASE FILE

The ERASE command enables the user to erase his own files, file generations, file generation groups or temporary files (data and catalog entries) and to release the space they occupy. He can also request a listing of the names of the erased files.

The command processes fully and partially qualified file names.

Operation	Operands
{ERASE ER}	{pathname * prefix *SYSLST *SYSLSTnn *SYSOUT *SYSOPT} [,{DESTROY DATA SPACE CATALOG}] [,LIST={N[O] Y[ES]}] [,POS={A[FTER] B[EFORE]}] [,VOLUME=vsn]

"pathname" stands for: {:catid:[$userid.][filename]
$userid.[filename]
filename}

catid — This operand denotes the catalog identifier associated with the file. If no entry is made, the default catalog ID assigned to the user ID is assumed.

userid — This operand denotes the user identification. If no entry is made, the user's own user ID is assumed. Only the user's own user ID is permitted.

filename — This operand specifies which files are to be erased. The specification can be a fully or partially qualified file name, or the name of a temporary file, of a file generation or of a file generation group.

If the name of a file generation is specified, the POS operand must also be specified.

* — This operand refers to the job's EAM object module file, which is created and used by the language processor. All other operands of the ERASE command are ignored.

prefix

This operand denotes a special character used a prefix (cf. section 2.1.1). The character is defined via the class 2 option at system generation time. If "YES" is entered in response to the message "DMSD516 ERASE ALL FILES S.TMP.nnnn.? REPLY (Y/N)", all temporary files are deleted.

*SYSLST
*SYSLSTnn
*SYSOUT
*SYSOPT

These system files can be erased (00≤nn≤99). In the case of *SYSOUT and *SYSOPT, the ERASE command is rejected if the system file to be erased does not have its primary allocation.

Notes:

- If the *SYSLST, *SYSLSTnn, *SYSOUT or *SYSOPT operand is specified, all other operands in the ERASE command are ignored.
- The operand *SYSOUT is applicable in batch mode only. If an ER *SYSOUT is followed immediately by a LOGOFF command, no new SYSOUT file is created for the LOGOFF listing.

DESTROY

This operand specifies that the space occupied by the file "filename" is released, the catalog entry of the file(s) deleted and the data overwritten with binary zeros.

DATA

This operand specifies that the data in "filename" is to be logically deleted, i.e. the user will no longer be able to reference the data since he is not allowed physical access to volumes.

The individual file remains cataloged and the space allocated.

In this case, the form of the catalog entry is identical to the entry made via a FILE command for a file that has not been opened before.

SPACE

This operand releases the space allocated to "filename". The individual file remains cataloged, but is treated as logically deleted.

This operand is not valid for private disks, and is rejected.

CATALOG

This operand specifies that the catalog entry for "filename" is to be deleted.

The CATALOG operand can be used only for files, file generations or file generation groups on private volumes.

LIST=N[O]

The names of the deleted files are not listed.

=Y[ES]

This operand specifies that the names of the deleted files are to be written in alphabetical order to SYSOUT.

POS=

This operand is effective only when "filename" is the name of a file generation; it causes deletion of one or more generations. The generation number of the oldest (FIRST) and/or the most recent file generation (LASTGN) is adapted to the altered situation, if applicable.

=A[FTER] This operand specifies deletion starting at the relative or absolute generation number specified in "filename". The generation specified by "filename" must exist, and is not deleted.

=B[EFORE] This operand specifies deletion as far as the relative or absolute generation number specified in "filename". The generation specified by "filename" must exist, and is not deleted.

VOLUME=vsn This operand is valid only for private volumes in conjunction with the operand CATALOG.

It deletes all catalog entries for files whose catalog entries contain an entry referring to this volume. A function with the opposite effect is contained in the IMPORT command.

When a volume containing generations of a file generation group is exported, gaps may arise in the file generation group; refer to the notes for the IMPORT command.

Note:
A file name entry in these operands is optional.

Notes:

- If the operands DESTROY, DATA, SPACE and CATALOG are not specified in the ERASE command, the space allocated to the file is released and its catalog entry deleted.

- For files with the entry ACCESS=READ in the catalog (see the CATALOG command), only the catalog entry can be deleted, i.e. in the ERASE command only the CATALOG operand can be specified.

- If a file on a private disk is to be deleted, the device on which that disk is mounted is requested for the job. After execution of the ERASE command, the device is returned to the system.

- For files on private disks, the following volumes must be mounted at command execution time:

  - all volumes belonging to the file, if the DESTROY operand is specified in the ERASE command;

  - the first volume, if the DATA operand is specified;

  - none of the volumes belonging to the file, if the CATALOG operand is specified in the ERASE command.

  These rules also apply if several files are referenced in the ERASE command via a partially qualified file name. In this case, the volumes for all the files concerned need not be mounted simultaneously. The system is requested to provide no more than the number of devices that are required for the file occupying the most volumes.

- If an error occurs during deletion of a file generation group or a number of file generations in a group, the remaining file generations are not deleted, and the catalog entry for the file generation group is updated.

- In a procedure file there is no need for a STEP command to follow erasure of the object module file (/ERASE *), since the ERASE command does not produce an error message if this file is empty.

- If a file does not exist, the DMS sends a message to the symbolic system file SYSOUT, and continues by processing the next command. In DO procedures a branch is made to the next STEP command.

- Although S.IN... files generated by the system are password-protected (EXPASS), they can still be erased by means of the ERASE command without first specifying the password. In this way S.IN... files that are no longer required or have not been erased by the system can be removed from the system.

- In addition to the catalog entry on the private disk, a private file can have a corresponding catalog entry on one or more pubsets. However, when the ERASE command/macro is executed, only the catalog entry on the pubset referenced by the catalog identifier (explicit specification or default catalog ID) is erased, i.e. catalog entries for the now erased file continue to exist on other pubsets.

<u>Network catalog management (see also the "MSCF Multiprocessor System" manual)</u>

The user can specify only his own user ID, but any valid catalog identifier.

Example 1:

The following listing is produced in interactive mode:

```
/FSTAT M.PROC.4
0000087 :V:$PM211034.M.PROC.4
:V:  PUBLIC:       1 FILE.  RES=        87, FREE=        1, REL=        0 PAGES
/ER M.PROC.4  ------------------------------------------------------------ 1)
/FSTAT M.PROC.4
%  DMS0533 REQUESTED FILE NOT CATALOGED ON PVS V. CMD TERMINATED
/ER M.PROC.,LIST=YES  ---------------------------------------------------- 2)
%  DMS0516 ERASE ALL FILES :V:$PM211034.M.PROC.? REPLY (Y=YES; N=NO)?
Y
%  DMS0800 ERASE FILE :V:$PM211034.M.PROC.ENDP
%  DMS0800 ERASE FILE :V:$PM211034.M.PROC.ENDP-SKIP
%  DMS0800 ERASE FILE :V:$PM211034.M.PROC.NAME
%  DMS0800 ERASE FILE :V:$PM211034.M.PROC.3
%  DMS0800 ERASE FILE :V:$PM211034.M.PROC.3.1
%  DMS0800 ERASE FILE :V:$PM211034.M.PROC.3.2
/FSTAT M.PRIM.FOR,S
0000087 :V:$PM211034.M.PRIM.FOR
  FCBTYPE = ISAM     VSNTYPE = PUB        LASTPG  = 0000086   2ND ALLO= 00012
:V:  PUBLIC:       1 FILE.  RES=        87, FREE=        1, REL=        0 PAGES
/ER M.PRIM.FOR,DATA  ----------------------------------------------------- 3)
/FS M.PRIM.FOR,S
0000087 :V:$PM211034.M.PRIM.FOR
  FCBTYPE = NONE     VSNTYPE = PUB        LASTPG  = 0000000   2ND ALLO= 00012
:V:  PUBLIC:       1 FILE.  RES=        87, FREE=       87, REL=       87 PAGES
/ER M.  ------------------------------------------------------------------ 4)
%  DMS0516 ERASE ALL FILES :V:$PM211034.M.? REPLY (Y=YES; N=NO)?
Y
%  DMS0801 ERASE ERROR ON FILE :V:$PM211034.M.KOP.SAM
%  DMS05C3 FILE TO BE ERASED IS IN USE. COMMAND TERMINATED
/FSTAT M.
0000051 :V:$PM211034.M.KOP.SAM
:V:  PUBLIC:       1 FILE.  RES=        51, FREE=        1, REL=        0 PAGES

/FSTAT DGG1,GEN=YES
0000000 :V:$PM211023.DGG1 (FGG)
0000006 :V:$PM211023.DGG1(*0001)
0000009 :V:$PM211023.DGG1(*0002)
0000012 :V:$PM211023.DGG1(*0003)
0000012 :V:$PM211023.DGG1(*0004)
0000006 :V:$PM211023.DGG1(*0005)
:V:  PUBLIC:       6 FILES. RES=        45, FREE=        4, REL=        0 PAGES
/ER DGG1(*3),POS=BEFORE  ------------------------------------------------- 5)
/FSTAT DGG1,GEN=YES
0000000 :V:$PM211023.DGG1 (FGG)
0000012 :V:$PM211023.DGG1(*0003)
0000012 :V:$PM211023.DGG1(*0004)
0000006 :V:$PM211023.DGG1(*0005)
:V:  PUBLIC:       4 FILES. RES=        30, FREE=        3, REL=        0 PAGES
/ER DGG1(*3),POS=AFTER  -------------------------------------------------- 6)
/FSTAT DGG1,GEN=YES
0000000 :V:$PM211023.DGG1 (FGG)
0000012 :V:$PM211023.DGG1(*0003)
:V:  PUBLIC:       2 FILES. RES=        12, FREE=        1, REL=        0 PAGES
```

1) The file M.PROC.4 is deleted; its storage space (3 PAM blocks) is released and its catalog entry deleted.

2) A partially qualified name has been specified in this ERASE command, with the result that several files, namely all those whose names begin with "M.PROC.", are deleted. A list of these files is displayed on the terminal (SYSOUT).

3) The data in file M.PRIM.FOR is deleted, but the file's storage space is not released, as is shown by the subsequent FSTATUS command.

4) An error message is issued for files which cannot be deleted. These two files cannot be deleted until:

   1. the access authorization for the file M.KOP.SAM is changed to ACCESS=WRITE:

      /CATALOG M.KOP.SAM,STATE=UPDATE,ACCESS=WRITE

   2. the password required for the file M.VORH is entered with the PASSWORD command.

5) All file generations prior to DGG1(*3) are deleted. DGG1(*3) is now the oldest generation of file generation group DGG.

6) All file generations subsequent to DGG1(*3) are deleted. DGG1(*3) thus becomes the latest file generation of DGG.

Example 2:

The command "ERASE A." is to reference two files:
file A.1, which is resident on two private volumes, and file A.2, which is resident on three private volumes. The ERASE command then requests three devices (not 5), since for file A.1 two devices, and then for file A.2 three devices need to be mounted simultaneously.

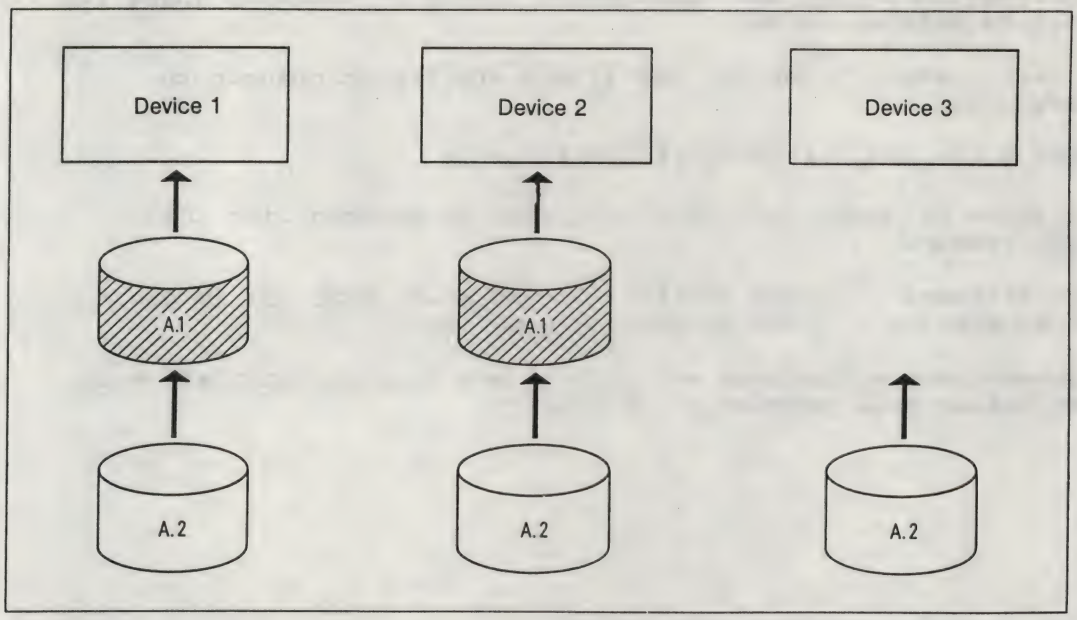


Fig. ER-1: Deleting files on private volumes

In the case of deletion accompanied by the release of storage space, the number of devices required is thus equal to the number of volumes belonging to the file occupying the most volumes.

FILE DEFINE FILE ATTRIBUTES

The file command is used to define files (and file generations) and their attributes. All files except EAM files must first be defined by this command (or the FILE macro). The file is thereby cataloged and the specified attributes then remain known to the system. There is therefore no need to repeat these attributes in a FILE command when the file is to be referenced again.

Principal functions of the FILE command:

- Catalog a new file or file generation (see the "filename" operand).
- Generate an entry in the Task File Table TFT (see the LINK operand and Fig. FILE-1).
- Request devices and volumes (see the operands DEVICE and VOLUME).
- Cause console messages requesting the mounting of private volumes (see the MOUNT operand).
- Allocate and deallocate storage space on disks (see the SPACE operand).
- Allow processing of non-cataloged files or file generations on private volumes (see the STATE=FOREIGN operand).
- Provide information regarding data organization on tapes (see the LABEL, TPMARK, CODE operands).
- Specify a file retention period (see the RETPD operand).
- Specify the "OPEN mode" (see the OPEN operand).
- Specify the data structure for the various access methods.

Operands	Access method: SAM	ISAM	PAM
FCBTYPE	X	X	X
RECFORM	X	X	i
RECSIZE	X	X	i
BLKSIZE	X	X	i
SHARUPD		X	X
KEYLEN		X	i
KEYPOS		X	i
DUPEKY		X	i
LOGLEN		X	i
VALLEN		X	i
VALPROP		X	i
OVERLAP		X	i
PAD		X	i
WROUT		X	i

where:

i: ignored but not rejected, since in certain cases ISAM or SAM files can be accessed with PAM (see section 6.2).

- Request private devices and volumes, and also storage space for the file data section of ISAM files if this is to be separated from the file index section (see the operands DDEVICE, DVOLUME, DSPACE).

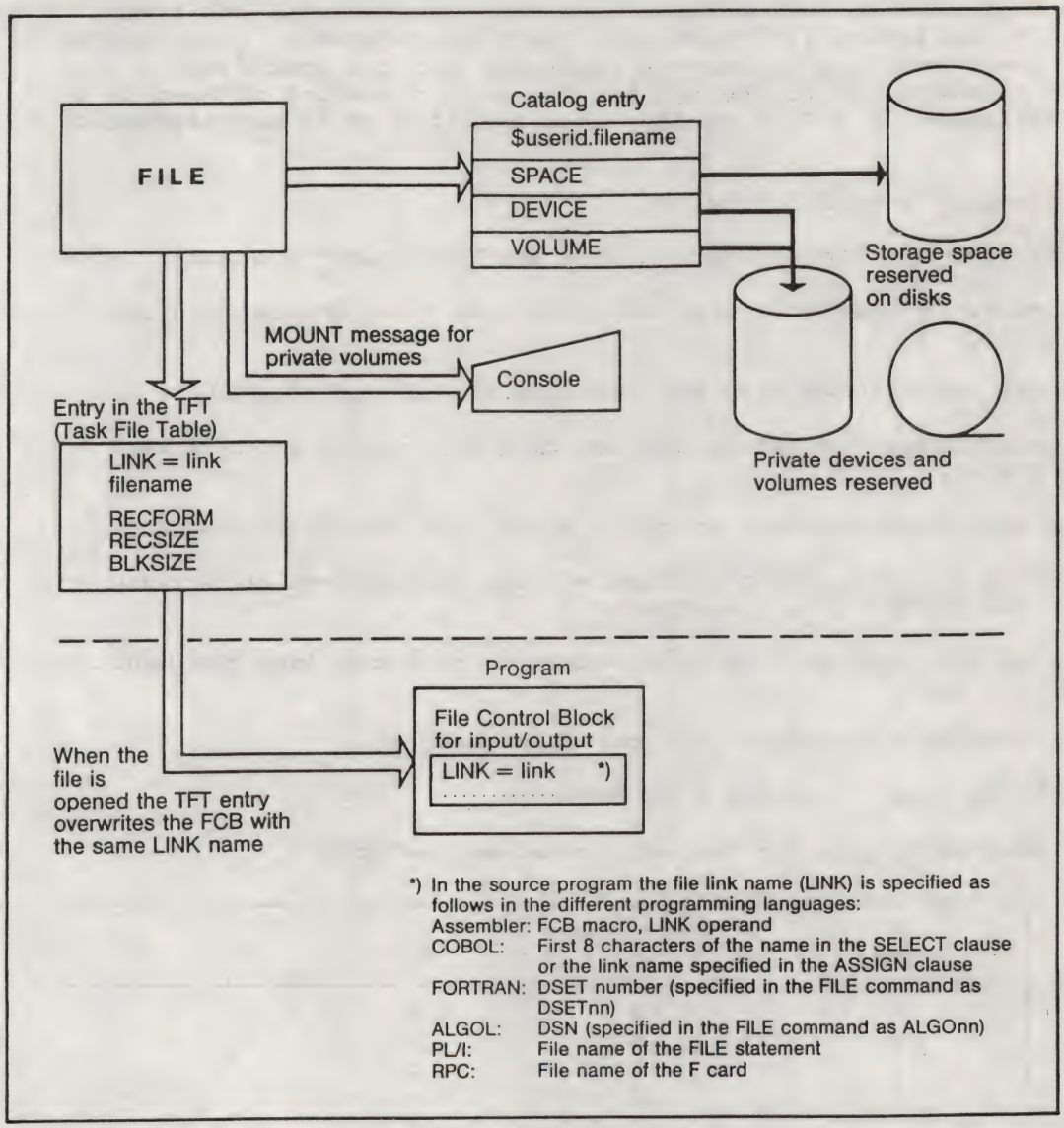


Fig. FILE-1: Functions of the FILE command

FILE DEFINE FILE ATTRIBUTES

The file command is used to define files (and file generations) and their attributes. All files except EAM files must first be defined by this command (or the FILE macro). The file is thereby cataloged and the specified attributes then remain known to the system. There is therefore no need to repeat these attributes in a FILE command when the file is to be referenced again.

Principal functions of the FILE command:

- Catalog a new file or file generation (see the "filename" operand).
- Generate an entry in the Task File Table TFT (see the LINK operand and Fig. FILE-1).
- Request devices and volumes (see the operands DEVICE and VOLUME).
- Cause console messages requesting the mounting of private volumes (see the MOUNT operand).
- Allocate and deallocate storage space on disks (see the SPACE operand).
- Allow processing of non-cataloged files or file generations on private volumes (see the STATE=FOREIGN operand).
- Provide information regarding data organization on tapes (see the LABEL, TPMARK, CODE operands).
- Specify a file retention period (see the RETPD operand).
- Specify the "OPEN mode" (see the OPEN operand).
- Specify the data structure for the various access methods.

Operands	Access method: SAM	ISAM	PAM
FCBTYPE	X	X	X
RECFORM	X	X	i
RECSIZE	X	X	i
BLKSIZE	X	X	i
SHARUPD		X	X
KEYLEN		X	i
KEYPOS		X	i
DUPEKY		X	i
LOGLEN		X	i
VALLEN		X	i
VALPROP		X	i
OVERLAP		X	i
PAD		X	i
WROUT		X	i

where:

i: ignored but not rejected, since in certain cases ISAM or SAM files can be accessed with PAM (see section 6.2).

- Request private devices and volumes, and also storage space for the file data section of ISAM files if this is to be separated from the file index section (see the operands DDEVICE, DVOLUME, DSPACE).

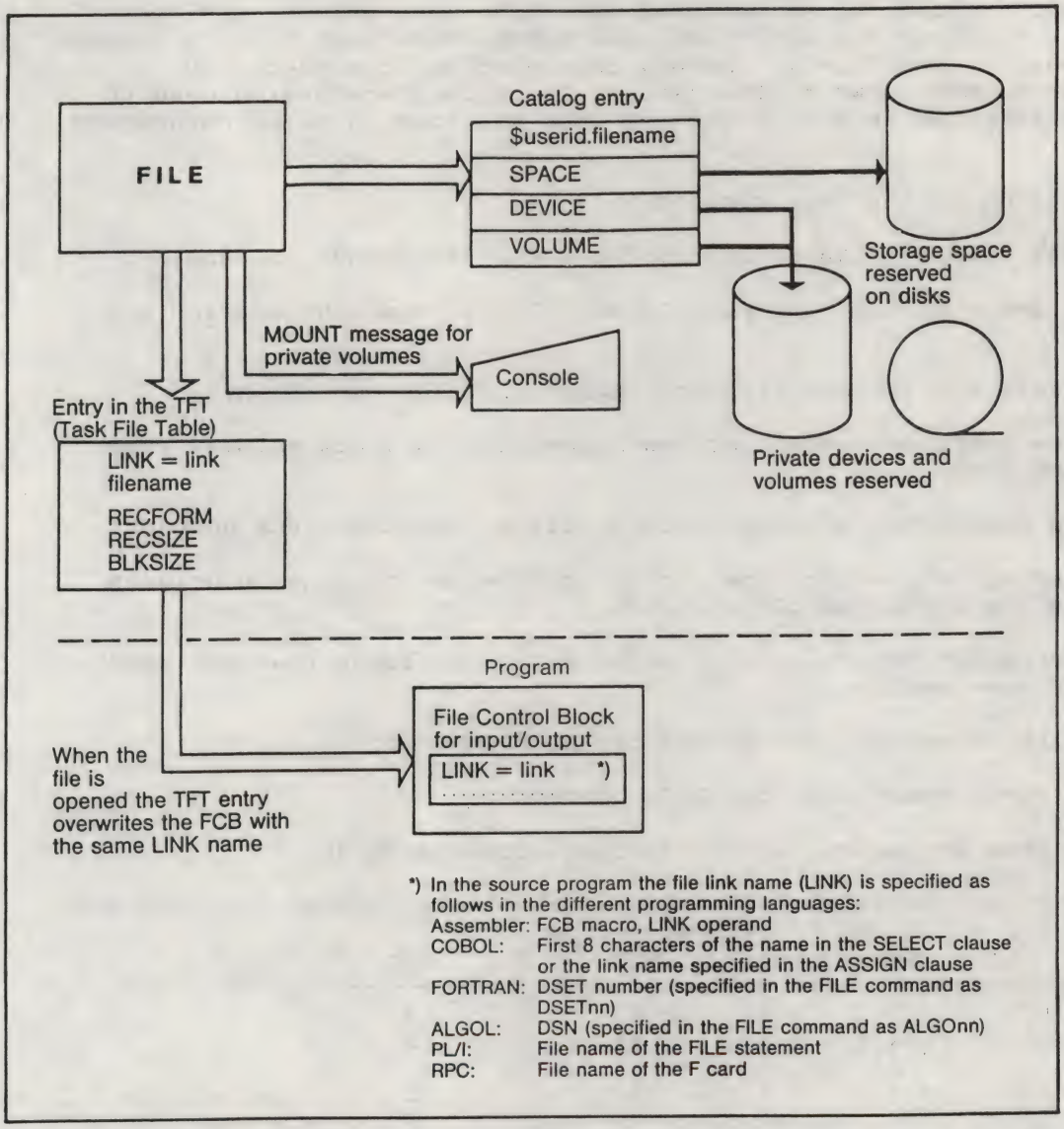


Fig. FILE-1: Functions of the FILE command

Explanation of Fig. FILE-1:

During processing of a FILE command, the system performs a number of functions, e.g.:

- It creates a catalog entry for the specified file name, or, if the file has already been cataloged, uses the existing catalog entry. The system then transfers to this catalog entry the results of the device, volume and space reservations, which can be controlled via the DEVICE, VOLUME and SPACE operands. Entries in all other operands are ineffective if no LINK name is specified.

- It makes an entry in the Task File Table (TFT) for the specified file link name (LINK name). This entry includes the file characteristics, namely the values of the operands LINK, "filename", RETPD, OPEN, RECFORM, RECSIZE, BLKSIZE, KEYPOS, KEYLEN, PAD, LOGLEN, VALLEN, VALPROP etc., but only those which were explicitly specified.

The file characteristics in the TFT are not automatically transferred to the catalog entry upon processing of the FILE command. The TFT entries can take effect when the associated file is opened only if a program uses the same file link name as the FILE command. When the file is opened, the system collects the file characteristics from three different sources:

1. from the program's file definition (FCB);

2. from the TFT entry with the corresponding file link name provided by the FILE command;

3. from the catalog entry created by the CATALOG, COPY, DATA, FILE, SYSFILE, or VERIFY command, or, in the case of preceding file processing, via a program.

The specifications in the TFT entry, i.e. everything defined explicitly via a FILE command, overwrite any corresponding FCB entries which may exist when the file is opened. Entries relating to file characteristics which are in neither the FCB nor the TFT entry are then transferred from the file's catalog entry.

Note:

For a file which already contains the appropriate information in its catalog entry, the operands FCBTYPE, RECFORM, RECSIZE, BLKSIZE, KEYLEN, LOGLEN, VALLEN and VALPROP can be specified in the form of "null operands", e.g. FILE..., FCBTYPE=, RECFORM=, RECSIZE=,.... This ensures that, when the file is opened, this data is always taken from the file's catalog entry, i.e. entries in the FCB have no effect.

Format for disk files

Operation	Operands
FILE	[{pathname | *DUMMY}]
	[,LINK=link]
	[,DEV[ICE]=device]
	[,VOL[UME]={PRIVATE | (PRIVATE,number) | vsn | (vsn,...)}]
	[,MOUNT=seqno]
	[,SPACE=[{primary | (primary[,secondary]) | (firstpage,amount,ABS)}]]
	[,STATE=FOREIGN]
	[,RETPD=days]
	[,OPEN={INPUT | OUTPUT | EXTEND | REVERSE | UPDATE | OUTIN | INOUT}]
	[,FCBTYPE={ISAM | SAM | PAM}]
	[,RECFORM={{V | F | U} | ({F | V | U}[,{N | M | A}])}]
	[,RECSIZE=reclength]
	[,BLKSIZE={STD | (STD,integer)}]
	[,SHARUPD={NO | YES}]
	[,KEYPOS=displacement]
	[,KEYLEN=keylength]

Operation	Operands
	[,DUPEKY=YES]
	[,LOGLEN=flag1]
	[,VALLEN=flag2]
	[,VALPROP={MIN | MAX}]
	[,OVERLAP=YES]
	[,PAD=percent]
	[,WROUT={NO | YES}]
	[,DDEVICE=ddevice]
	[,DVOLUME={vsn | (vsn,...)}]
	[,DSPACE={primary | (primary[,secondary]) | firstpage,amount,ABS)}]
	[,WRCHK={YES | NO}]

"pathname" stands for: [:catid:][$userid.]filename

catid — This operand denotes the catalog identifier associated with the file. If no entry is made, the default catalog ID assigned to the user ID is assumed.

userid — This operand denotes the user identification associated with the file. If the operand is omitted, the user's own user ID is assumed.

filename — This operand specifies the fully qualified name of a file or the name of a file generation or of a temporary file, to which all the other operands in the FILE command refer. The name of a file generation group must not be specified.

If, at this stage, the file or file generation has not yet been cataloged, this is now performed.

*DUMMY — This operand defines the dummy file. This specification allows input and output operations to be simulated in the program.

1. Input: If a program attempts to read this dummy file, the program's end-of-file routine is activated, i.e. as though the file had already been read.

2. Output: If an attempt is made to write to the dummy file, the data is still transferred to the program's buffer areas (IOAREA), but buffer output to a volume does not take place.

If the DUMMY operand is used, all the other FILE command operands -- except LINK -- are checked for syntax errors but are otherwise ignored. Therefore no device assignment, storage space allocation or cataloging is performed for the dummy file.

This procedure can be used for program test runs, or if an output is to be temporarily skipped.

LINK=link

This operand specifies a file link name not exceeding 8 bytes. The assignment of link names is governed by the same rules as for the use of file names (cf. section 2.1.2). If omitted, no TFT entry is made. An entry is made under this name in the Task File Table (TFT). If the link name already has a TFT entry at this time, the latter is replaced by the new TFT entry. If devices were connected with the old TFT entry, this connection is now dissolved, though the devices are still available to the job.

DEVICE=device

This operand specifies the device type. The following can be entered for "device":

Device	For disk storage unit	Device code
D3455 D5881	3455	A8
D3465 D5882	3465	A9
D3468	3468	AA
D3470 D5804	3470	A6
D3475	3475	AB
D3480	3480	AC
D348E	348E	AD

Default value:
If the DEVICE and VOLUME operands are omitted from the FILE command, a device for a public volume is assumed.

VOLUME=PRIVATE This operand specifies that the operator must mount private volumes. These volumes include those disks which do not have to be permanently connected to the system, i.e. are not public volumes. As soon as a private volume is assigned, it belongs to the user.

=(PRIVATE,number)
This operand defines the desired number of private volumes, where "number" is an integer in the range 1 through 9.

=vsn This operand specifies the volume serial number (not exceeding 6 bytes) which serves to identify volumes. A maximum of 100 volume serial numbers can be specified, separated from each other by commas and enclosed in parentheses.

With **disk files**, the system attempts to satisfy the space requests (see the SPACE operand) on the first volume specified. If this is successful, the other volume serial numbers specified are also entered in the catalog (for future extensions of the file).

Note:

If "pathname" is the name of a **file generation** on private volumes, and DISP=REUSE (see the CATALOG command) was specified for the associated file generation group, the entries in the VOLUME operand are ignored when the maximum permissible number of cataloged file generations has been reached (see the GEN operand in the CATALOG command). If file generation "pathname" replaces an existing generation within its file generation group, it also adopts the latter's private volumes.

If a private file that is already mounted is to be extended to include another vsn, a primary allocation ≠ 0 must be simultaneously specified with SPACE. Otherwise, the FILE command will be ignored.

If a FILE command with the VOLUME or SPACE operand is specified for a file which is currently open, it will be rejected.

MOUNT=seqno This operand mounts the file's private volumes. A MOUNT message is displayed at the console.
For example, the sequence number 2 designates the second entry in the file's volume table, which may contain each vsn once only.

Default value: If the operand is omitted, the volume on which the file begins is requested.

MOUNT=0
A volume request is not carried out until the file is opened, provided it exists and the operands DEVICE, VOLUME, SPACE and DSPACE are not specified.

SPACE= This operand specifies changes in storage space allocation for disk files:

- the primary allocation "primary" stipulates how much space is to be allocated to the file initially;
- the secondary allocation "secondary" determines how much space is to be allocated automatically if additional space is required in the course of file processing;
- the absolute allocation "(firstpage,amount,ABS)" gives the user the additional option of specifying physical page numbers on the volume in the case of files on private disks.

Absolute and non-absolute allocation can be specified in any order for a private disk file.

If the SPACE operand is omitted, the system default allocation becomes effective (as defined at system generation time).

If a FILE command with the operand VOLUME or SPACE is specified for a file which is currently open, it will be rejected.

If a storage space request for public volumes causes the maximum value specified in the JOIN entry of the user ID to be exceeded, the request will be rejected or only partly honored (message DMS054A, but not if SPACE=ABS is specified). If the user ID is expressly authorized (enforcement) to exceed this maximum number of PAM blocks, however, there is merely a warning message.

Both primary and secondary allocation must be either greater than or equal to the specified buffer length (see the operand BLKSIZE).

In order to reduce system administration overheads, the following should be considered when selecting the values for primary and secondary allocations:

- primary allocation should correspond to the estimated size of the file to be set up.
- A minimum of 9 to 15 PAM blocks should be selected for secondary allocation.
- For large files the values for primary and secondary allocation should be selected as multiples of the management units packet (= 24 PAM blocks) or segment (= 192 PAM blocks).

=primary If necessary, the number "primary", the **primary allocation**, is rounded up to a multiple of 3.

Positive number specifications define the number of PAM blocks (2048 bytes) which are to be marked as reserved for the file (in addition to any space already reserved).

Negative number specifications can be made for files under the user's own user ID to which space has already been allocated. The system then attempts to release space. The greatest value that can be specified is 9999999.
Specifications in the VOLUME operand are ignored. The remaining file size is at the same time a multiple of 3 and of the buffer size.
Release of storage commences on the last volume entered in the catalog and continues until either the specified number of PAM blocks "primary" has been released, or a PAM block which contains data is encountered (applies to ISAM and SAM). Thus, deallocation starts at the end of the file and proceeds backwards towards the start of the file. ISAM file storage space cannot be released separately for the data and index sections (see DSPACE).
In the case of files on private disk, at least 3 PAM blocks remain allocated.

=(primary,secondary)

The **secondary allocation** ("secondary") specifies the number of PAM blocks (2048 bytes) to be reserved for the file whenever more storage space is required (dynamic allocation).
If necessary, this number will be rounded up to a multiple of 3 when the reservation takes place.
Maximum value for "secondary": 32767.

A **zero** primary allocation means that no further storage space is to be allocated with the FILE command. This specification is required, for example, if the user wishes to change only the value of the secondary allocation for the file but not the current space allocation. If "filename" does not yet exist, zero is not permitted as the primary allocation on private disks.

The secondary allocation for a file is recorded in its catalog entry and can be changed at any time.

A **zero** secondary allocation suppresses dynamic allocation. This specification remains effective until a new secondary allocation is specified.

Storage space requests made with SPACE=(primary, secondary) refer to the whole file, i.e. not to a particular volume.

1. Allocation of storage space for new files

If several volume serial numbers are specified in the VOLUME operand for a file for which space is being allocated for the first time, the system attempts to reserve all the space on the first volume.
If this is successful, the other volume serial numbers are entered in the catalog (for further extensions).

2. **Extension of storage space allocation for files on public disks**

   With these files, the system first attempts to allocate space on the last volume of the file.
   If this is inadequate, the system attempts to reserve the space on the volume with the most free space. (To be precise, the public disk with the highest percentage of free packets is selected, a packet being 24 contiguous PAM blocks.)

3. **Extension of storage space allocation for files on private disks**

   If the user specifies no volume serial numbers in the VOLUME operand, the system attempts to reserve space on the last volume of the file. If this is inadequate, the space is sought on other private disks entered in the catalog.
   If the user does specify volume serial numbers in the VOLUME operand, the system attempts to reserve space on the specified volumes. If this is inadequate, a partial assignment is made. Any space on disks whose volume serial numbers are entered in the catalog but do not appear in the current FILE command, is ignored.
   If a file, previously cataloged only on a private disk, is to extend over a second private disk, the operands VOLUME, DEVICE and SPACE must be specified. If the operand SPACE is not specified, the FILE command will be ineffective.

=(firstpage,amount,ABS)

This operand is accepted for private disks only.
This value for the SPACE operand is meaningful only if a vsn is specified in the VOLUME operand at the same time.

ABS specifies that an absolute allocation is to be made. If this is the first allocation for a file, the value of the secondary allocation is set to zero.

The entry **firstpage** specifies the physical block (2048 bytes) at which the allocation is to start. Since only multiples of 3 can ever be requested, the entry "firstpage" must contain the values 1, 4, 7, etc., i.e. the entry "firstpage" divided by three must always leave a remainder of 1.

The block with which reservation can begin is determined by disk initialization.

The entry **amount** defines the number of PAM blocks to be requested. This value must be a multiple of 3.

An absolute allocation is made only if the whole of the area requested can be reserved for the file, i.e. partial assignments are not possible.

In the case of absolute allocation, only one volume can be referenced per FILE command. If the user has specified more than one vsn in the VOLUME operand, only the first vsn is used. If space is to be allocated on several (n) volumes, the FILE command must then be issued n times.

Types of disk storage unit with specified capacities:

Disk Storage Unit (type)	3455 5881	3465 5882	3468	3470 5804	3475	3480	348E
Number of PAM blocks x)	30774	61473	130242	202908	183246	225663	451605

x) Data blocks without spare cylinder, standard labels and system-resistent blocks (SYSIPL).

STATE=FOREIGN

This operand specifies that an existing non-cataloged file on private volumes is to be cataloged. This file may have been:

- previously deleted from the catalog by the ERASE command (CATALOG operand) or
- created in a different BS2000 computer center, or
- created under a different operating system.

If "filename" is a file generation, it should be noted that the catalog entry for the associated file generation group must be present and thus, if necessary, must first be restored by means of the CATALOG command.
The following should be noted with regard to STATE=FOREIGN:

For files on private disks with the specification STATE=FOREIGN, the first vsn for the file will be specified and only the first volume for the file need be mounted when the file is opened. The system then takes the file attributes from the F1 label and enters them in the catalog. The F1 label contains the user ID of the file's creator, which must be present in the JOIN file of the operating system in order to allow cataloging of the FOREIGN file via the FILE command.

If the user issuing the FILE command for the FOREIGN file is not the creator of the file, he must specify in his FILE command the complete file name, including the user ID of the file's creator. In addition, the creator must, in this case, have cataloged the FOREIGN file as shareable (SHARE=YES in the CATALOG command), in order to permit access under another user ID.

Files on private volumes which, with the specification STATE=FOREIGN, are taken over into a different operating system or entered in a different user's catalog must be deleted from the catalog of the old system or of the owner (ERASE command with the CATALOG operand). For if the file is modified in the new system, and an attempt is subsequently made to access the file with the old catalog entry, problems may arise with the possible result that none of the files on the private volume can continue to be used. (This does not apply to shareable private disks.)

RETPD=days

This operand specifies the file retention period in days, during which time the file can be read only.
"days" must be a decimal integer, the maximum value of which is calculated from the difference in days between the end of the century (31.12.99) and the current date.
This retention period can be specified only for files to be cataloged for the first time.
When the retention period has expired, the file can be updated, but it is not automatically deleted.
RETPD>0 has no effect for temporary files.
Default value: zero days, i.e. the file can be updated immediately.

OPEN=

This operand specifies the mode in which the file is to be opened. This entry can, however, be overwritten by the operand in the OPEN macro when the file is opened.

No FILE command with the operands VOLUME and DEVICE can be given for files on private disk that are opened in INPUT mode.

The following table indicates the possible specifications for the OPEN operand in accordance with the various access methods for disk files:

OPEN =	SAM	ISAM	UPAM
INPUT	X	X	X
OUTPUT	X	X	-
EXTEND	X	X	-
REVERSE	X	-	-
UPDATE	X	-	-
INOUT	-	X	X
OUTIN	-	X	X

The individual OPEN modes are discussed under the corresponding access methods.

FCBTYPE={ISAM / SAM / PAM}

This operand specifies the access method with which the disk file is to be processed.

RECFORM=V This operand specifies variable-length records, i.e. each data record starts with a 4-byte record length field in which the record length is entered.

=F This operand specifies fixed-length records.

=U This operand specifies records of undefined length, i.e. the record length must be defined at input/output time. This is not allowed with the ISAM access method.

The following additional entries can be made:

"A" specifies that the first data byte is an ASA printer control character (print with SPACE=E).

"M" specifies that the first data byte will be interpreted as a control character in EBCDIC (print with SPACE=E).

"N" specifies that there is to be no printer control character (SPACE=E must not be used when printing out.

When using the PAM access method, the RECFORM operand can be specified but the system will ignore it.

RECSIZE=reclength

This operand specifies the length in bytes for fixed-length records (RECFORM=F).

For variable-length records (RECFORM=V), this operand specifies the maximum permissible length for a record, including its 4-byte control field. With variable-length records, it is not mandatory to specify RECSIZE; the system initially assumes the maximum possible record length is equal to the buffer length (BLKSIZE).
If the record length is undefined (RECFORM=U), the RECSIZE operand specifies one of the general-purpose registers 2 through 12, which is to contain the record length for input/output. As soon as a format-U record is read, the system supplies the general-purpose register with the length specification for this record. If such a record is to be written, the user must provide the length specification in the specified general-purpose register.

When using the PAM access method, the RECSIZE operand can be specified but the system will ignore it.

BLKSIZE=STD This operand specifies a 2048-byte buffer (one PAM block) for input/output involving the file. Data transfer to and from the I/O devices thus takes place in units of this size.

=(STD,integer)

This operand specifies that the buffer length is a multiple of 2048 bytes (one PAM block). "integer" defines the number of PAM pages and must not exceed 16, i.e. the maximum buffer size is 32768 bytes.

If the "integer" entry is greater than or equal to 2, the following applies:

- Primary allocation operand SPACE):
  "primary" ≥ (2 * "integer") for SAM files
  "primary" ≥ (2 * "integer") + 1 for ISAM files

Example:

BLKSIZE=(STD,8) --> "primary" ≥ 18 (SAM)
BLKSIZE=(STD,6) --> "primary" ≥ 15 (ISAM)

- Secondary allocation (operand SPACE):
  "secondary" ≥ "integer" for SAM and ISAM files

  Example:

  BLKSIZE=(STD,8) --> "secondary" ≥ 9

For UPAM files processed with chained I/O, both primary and secondary allocation must be as large as the required number of PAM blocks transferred with each I/O operation.

With ISAM files, a record must not exceed the length (BLKSIZE=4). In the case of all other file types, the buffer length BLKSIZE is the upper limit for the record length.

Notes:

- In the BLKSIZE operand, it is also possible to specify an absolute value. However, it should be noted that the use of standard blocks must be specified if the file was created with standard blocks or with a block size n*STD, as no internal conversion from absolute value to a multiple of standard is performed.

- In the case of SAM access and SETL processing:
  When blocking is extensive, the buffer can hold only a maximum of 255 records. The length of the positioning information is only 1 byte.

SHARUPD=YES
This operand specifies that an ISAM or PAM file can be simultaneously updated by more than one job, i.e. the file is not locked as soon as a job has opened the file in output mode. This results in an increased I/O rate.
The WROUT function (WROUT operand) is thus activated.

=NO
This operand specifies that this file can be used by more than one job simultaneously only if all jobs are operating in read mode.

KEYLEN=keylength
This operand specifies the record key length in bytes for an ISAM file. Each record in the file must have this key length. The key must not exceed 255 bytes.
Default value: 8 bytes.

KEYPOS=displacement
This operand specifies the position of the first character of the record key. The key must be in the same position in every record of the ISAM file. If variable-length records are specified (RECFORM=V), the 4-byte record length field is considered to be part of the record.
Default value: 5 for RECFORM=V, 1 for RECFORM=F.

DUPEKY=YES

This operand specifies that duplicate ISAM keys are allowed.

Default value: If duplicate keys are encountered, control is passed to a user program routine whose address was given in the program's EXLST macro.

Note:
The introduction of alternate keys will lead to restrictions with regard to duplicate keys. These restrictions will affect files for which alternate-key functions are to be used. Consequently, the generation of files with duplicate keys (DUPEKY=YES in the FILE command) should be avoided wherever possible.

LOGLEN=flag1

This operand specifies the length of the logical flag for ISAM files. This flag, along with the ISAM key and the value flag, can form part of the ISAM record index. The sum of the length entries in KEYLEN, LOGLEN and VALLEN must not exceed 255.
Default value: Zero.

VALLEN=flag2

This operand specifies the length of the value flag for ISAM files. This flag, along with the ISAM key and logical flag, can form part of the ISAM record index. The sum of the length entries in KEYLEN, LOGLEN and VALLEN must not exceed 255.
Default value: Zero.

VALPROP={MIN / MAX}

This operand specifies how the value flag is to be propagated in the index of ISAM files on all index levels (value propagation). This entry allows the user to search the file more rapidly if the majority of value flags with ascending ISAM keys are rising or falling.

Each index record contains the lowest (MIN) or highest (MAX) value flag for the next lower-level block to which it is pointing.

Note for LOGLEN, VALLEN and VALPROP:
In subsequent ISAM versions, the flag functions (logical and value flags) will be replaced by alternate keys (i.e. multiple indices corresponding to the ANS-COBOL functions). As a result, flag processing (the LOGLEN, VALLEN and VALPROP operands) will not be developed further and should be avoided if possible.

OVERLAP=YES

This operand specifies that buffered or overlapped read operations are to be performed for ISAM if more than one data buffer exists. This applies to the ISAM macros GET and GETR.

Default value: No attempt is made to perform overlapping I/O operations. Nevertheless, ISAM does use the second buffer area (IOAREA2), albeit as a work area.

PAD=percent

This operand specifies the percentage of the buffer length which is to be reserved for subsequent extensions (e.g. insertion or extension of records) during creation of an ISAM file.
Default value: 15% of the data space is reserved.

WROUT=YES

This operand specifies that the data and index buffers of an ISAM file are to be written back to the volume immediately after a modification, thereby enhancing file security. However, this is necessarily accompanied by an increase in I/O activities. The WROUT function becomes effective after the ISAM macros STORE, ELIM, INSRT, PUTX and PUT (with PUT, only when the buffer is full).

=NO

Writing back is not performed immediately, and possibly not until the file is closed.

If SHARUPD=YES is specified for an ISAM file, this automatically includes the WROUT function, i.e. the WROUT operand becomes redundant.

DDEVICE=ddevice
DVOLUME=vsn
DSPACE=(primary[,secondary])

While DEVICE, VOLUME and SPACE relate to the file index section of the ISAM file, these three operands relate to its file data section. In this way, it is possible to create an ISAM file whose data and index blocks reside on different private disks.

These operands are meaningful only when the ISAM file is created on a private disk. When associated with the operand FOREIGN, they will result in an error message.

The operands DDEVICE, DVOLUME and DSPACE are specified in exactly the same way as their corresponding operands DEVICE, VOLUME and SPACE.

Exceptions:

1. The "ddevice" entry for DDEVICE contains only the disk volumes of "device" in DEVICE.

2. With DSPACE, there is no default value for new files.

If a user wants to use one of these three operands for a file which as yet has no storage space, he must specify all three operands. If space has already been allocated for a file, only the DSPACE operand need be specified in the FILE command. DDEVICE and DVOLUME do not then need to be specified.

If an ISAM file has been created in this way, it is subsequently impossible to place data and index blocks on one and the same volume. To do this, the file must be copied to a second ISAM file whose storage space was allocated without the use of DDEVICE, DVOLUME and DSPACE.

It is not possible to divide up an ISAM file of this kind on public volumes or on a mixture of private and public volumes.

DSPACE=(firstpage,amount,ABS)

This operand is accepted for private disks only; it is meaningless unless a vsn is specified at the same time in the VOLUME operand.

"ABS" specifies an absolute allocation. If this is the first allocation for a file, the value of the secondary allocation is set to zero.

The specification "firstpage" defines the physical block (2048 bytes) with which the space allocation is to begin. Since only multiples of 3 can ever be requested, the entry "firstpage" must contain the values 1, 4, 7, etc., i.e. "firstpage" divided by 3 must always leave a remainder of 1.

The entry "amount" defines the number of PAM blocks to be requested. This value must be a multiple of 3.

An absolute allocation is made only if the whole of the area requested can be reserved for the file, i.e. partial assignments are not possible.

In the case of absolute allocation, only one volume can be referenced per FILE command. If the user has specified more than one vsn in the DVOLUME operand, only the first vsn is used. If space is to be allocated on several (n) volumes, the FILE command must then be issued n times.

Note for DDEVICE, DVOLUME and DSPACE:
In subsequent ISAM versions, an improved buffer management will make the division into data section and file section redundant. For reasons of compatibility the DDEVICE, DVOLUME and DSPACE operands will still be supported, but they will have no effect on processing.

WRCHK=YES

A read-after-write check is performed on the data. This check refers simply to the checking of the records to ensure that the data written is readable; the contents are not checked. This method permits recording errors on the disk to be detected in good time and to be corrected by means of appropriate recovery measures. If the error cannot be recovered, a branch is made to the ERRADDR routine (via EXLST).

Notes:

- A read-after-write check requires an additional turning of the disk, which has an adverse effect on system performance.
- This entry is not recorded in the catalog entry of the file. This means that the entry must be specified explicitly for any processing of the file. It must be valid for the file when it is opened.

=<u>NO</u>

No read-after-write check is performed.

Example 1:

```
/FSTAT M.1
%  DMS0533 REQUESTED FILE NOT CATALOGED ON PVS V. CMD TERMINATED
/FILE M.1  ------------------------------------------------------------------ 1)
/FSTAT M.1,ALL
0000003 :V:$PM211034.M.1
  FCBTYPE = NONE     VSNTYPE = PUB       LASTPG  = 0000000   2ND ALLO= 00009
  SHARE   = NO       ACCESS  = WRITE
  ACCESS# = 000      CRDATE  = NONE      EXDATE  = NONE      LADATE  = NONE
  RDPASS  = NONE     WRPASS  = NONE      EXPASS  = NONE
  VERSION = 000      BACKUP# = 000       LARGE   = NO        BACKUP  = A
  DESTROY = NO       AUDIT   = NONE
  BLKTYPE = NONE     BLKSIZE = 000000    RECFORM = NONE      RECSIZE = 00000
  VSN/DEV/EXT =      PUBV09/D3475/001
  EXTCNT  =   1
:V:  PUBLIC:       1 FILE.  RES=        3, FREE=        3, REL=        3 PAGES

/FSTAT D.DGG1,TYPE=FGG,GEN=YES  ---------------------------------------------- 2)
%  DMS0533 REQUESTED FILE NOT CATALOGED ON PVS V. CMD TERMINATED
/CAT D.DGG1,GEN=5
/FILE D.DGG1(*1)  ------------------------------------------------------------ 3)
/FSTAT D.,TYPE=FGG,GEN=YES
0000000 :V:$PM211023.D.DGG1 (FGG)
0000003 :V:$PM211023.D.DGG1(*0001)
:V:  PUBLIC:       2 FILES. RES=        3, FREE=        3, REL=        3 PAGES
```

1) and 3) As a result of the FILE command, file M.1 and file generation D.DGG(*1) are:

- cataloged
- given a reservation of 3 PAM blocks (default value) on the public disk mounted on device type D3475 with the volume serial number PUBV09.

2) If there is as yet no group entry for D.DGG(*1), it must be created before the FILE command.

Example 2:

The file editor EDT performs standard processing on variable-length records. The file link name for the input/output of SAM files is EDTSAM.

In order, for example, to process fixed-length records (in the example the record length is 10 bytes), a FILE command must be issued before the call.

```
/FILE M.EDT,LINK=EDTSAM,RECFORM=F,RECSIZE=10
/EXEC EDT
%  BLS0500 PROGRAM EDT, VERSION 160 OF 84-12-03 LOADED.
```

Example 3:

In FORTRAN programs (also COBOL, RPG2, etc.) file link names can be specified (LINK=). A file name is not assigned to this link name until a FILE command is issued.

```
.
.
.
/FILE FOR.AUSGABE,LINK=DSET20
/EXEC FOR.PROGRAMM
/RELEASE DSET20
.
.
.
```

In this case, the FORTRAN program FOR.PROGRAMM contains the entry "20", i.e. the file link name DSET20. The FILE command connects the file name FOR.AUSGABE to this link name, i.e. a corresponding Task File Table (TFT) entry is set up which brings about the desired assignment during file opening.

Example 4:

Space reservation on public volumes (default)

```
/FILE M.2,SPACE=5 ------------------------------------------------------ 1)
/FSTAT M.2
0000006 :V:$PM211034.M.2
:V:  PUBLIC:      1 FILE.  RES=        6, FREE=        6, REL=        6 PAGES
/FILE M.2,SPACE=(15) --------------------------------------------------- 2)
/FSTAT M.2
0000021 :V:$PM211034.M.2
:V:  PUBLIC:      1 FILE.  RES=       21, FREE=       21, REL=       21 PAGES
/FILE M.2,SPACE=(-10) -------------------------------------------------- 3)
/FSTAT M.2
0000009 :V:$PM211034.M.2
:V:  PUBLIC:      1 FILE.  RES=        9, FREE=        9, REL=        9 PAGES
```

1) The space request for 5 PAM blocks is rounded up to a multiple of 3, with the result that a primary allocation of 6 PAM blocks is made.

2) An additional 15 PAM blocks are reserved.

3) Space is released. Here, too, the entry is rounded up to a multiple of 3.

```
/FILE M.3,SPACE=(9,9) -------------------------------------------------- 4)
/FSTAT M.3,S
0000009 :V:$PM211034.M.3
  FCBTYPE = NONE    VSNTYPE = PUB      LASTPG  = 0000000   2ND ALLO= 00009
:V:  PUBLIC:      1 FILE.  RES=        9, FREE=        9, REL=        9 PAGES
/FILE M.3,SPACE=(0,4) -------------------------------------------------- 5)
/FSTAT M.3,S
0000009 :V:$PM211034.M.3
  FCBTYPE = NONE    VSNTYPE = PUB      LASTPG  = 0000000   2ND ALLO= 00004
:V:  PUBLIC:      1 FILE.  RES=        9, FREE=        9, REL=        9 PAGES
```

4) Primary allocation and secondary allocation are specified.

5) Only the secondary allocation is changed.

FSTATUS REQUEST CATALOG INFORMATION

The FSTATUS command is used to request information from the catalog about the status of files. The user can request information for one or more individual files, for file generations, for file generation groups, for temporary files, or for all the files under one user ID. In addition, the information can be selected according to file attributes or file and volume type, or be restricted to newly opened file generations.

Operation	Operands
{FSTATUS | FSTAT}	[{pathname | prefix}]
	[{[,S[TANDARD]][,C[ATALOG]] [,T[RAITS]][,P[ASSWORD]] | [,ALL] | ,R[ESERVED]}]
	[,A[CCESS]={R[EAD] | W[RITE]}]
	[,B[ACKUP]={A | B | C | D | E | ({A | B | C | D | E},...)}]
	[,CR[DATE]={date | (date,) | (,date) | (date,date)}]
	[,EX[DATE]={date | (date,) | (,date) | (date,date)}]
	[,EXT[ENTS]={value | (value,) | (,value) | (value,value)}]

Operation	Operands
	[,FCB[TYPE]={P[AM] | S[AM] | I[SAM] | N[ONE] | (P[AM] | S[AM] | I[SAM] | N[ONE],...)}]
	[,F[REE]SIZE={SIZE | value | (value,) | (,value) | (value,value)}]
	[,FROM={CAT[ALOG] | (vsn,device)}]
	[,LA[DATE]={date | (date,) | (,date) | (date,date)}]
	[,L[IST]={target | (target,type}]
	[,P[ASS]={R[DPASS] | W[RPASS] | E[XPASS] | N[ONE] | (R[DPASS] | W[RPASS] | E[XPASS] | N[ONE],...)}]
	[,SAVE={Y[ES] | N[O]}]
	[,SH[ARE]={Y[ES] | N[O]}]
	[,SIZE={F[REE]SIZE | value | (value,) | (,value) | (value,value)}]
	[,SORT={F[ILE]NAM | N[O]}]

Operation	Operands
	[,STATE={NOCLOS | PCLOSE}]
	[,SUP[PORT]={PUB[LIC] | PRD[ISC] | (PUB[LIC],PRD[ISC])}]
	[,GEN={Y[ES] | N[O]}]
	[,TYPE=FGG]
	[,VTOC={Y[ES] | N[O]}]

"pathname" stands for: {:catid:[$userid.][filename] | $userid.[filename] | filename}

catid

This operand denotes a catalog identifier. If no entry is made, the default value assigned to the user ID is assumed.

Within the operands "catid" and "filename" the following symbols are allowed:

Symbol	Meaning
*	This replaces any (also a blank) character string.
/	This replaces any single character.
<s1:s2>	This replaces a character string situated lexicographically between the character strings "s1" and "s2" (both inclusive). Strings "s1" and "s2" may also be the blank strings. If string "s1" is shorter than string "s2", it will be padded with X'00' to match the length of string "s2". If character string "s2" is shorter than character string "s1", it will be filled with X'FF' to match the size of character string "s1".
<s1,...>	This replaces one of the specified character strings, which may also be a blank character string. Any of the specified alternatives may also consist of range "s1:s2" (see above).
-	This character is allowed only as a symbol at the beginning of a file name. It serves to negate the next specification, i.e. all files that do not satisfy the next name specification are listed.

userid

This operand specifies a user identification. The entry $userid designates all files belonging to the user. If the user ID of another user is specified, only information concerning shareable files (SHARE=YES) is output.
If the operand is not specified, the user's own user ID is assumed.

filename

This operand specifies the file(s) whose status is to be output. This may be a partially or fully qualified file name, the name of a file generation or the partially or fully qualified name of a file generation group.

The maximum length of the file name is 80 characters. Temporary files are also taken into account.

prefix

The prefix (special character) specifies that information on all the temporary files of this job is to be output. The internal name of these files is always output.

S[TANDARD]

This operand causes the following information to be output in addition to the current data on space allocation:

FCBTYPE	Access method used when the file was created.
VSNTYPE	Type of volume on which the file resides (PUB for public, PVT for private volumes).
2ND ALLO	Value of the secondary allocation for the file.
LAST PG	Number of the last page used by this file (only for files with standard blocks).

2nd Line:

:x:	identifier of the catalog ("catid") in which the files are cataloged.
FILES	Number of files on public volumes (PUBLIC), private disks (PRIVATE) and on tape (TAPE).
RES=	Total number of reserved PAM blocks.
FREE=	Number of free (not reserved) PAM blocks.
REL=	Number of PAM blocks that can be released. This is not the same as the number of free blocks, as PAM blocks can only be released in multiples of 3 for each file (cf. the FILE command).

C[ATALOG]

In addition to information on current space allocation, this operand outputs the following information defined by the CATALOG command.

SHARE	Shareability.
ACCESS	Type of file access (read only or read and write).
ACCESS#	Number of accesses. This count is set to 0 when the file is created and incremented by 1 each time the file is opened. Once the value 255 is reached, the count remains unchanged.
CRDATE	Creation date (YY-MM-DD).
EXDATE	Expiration date (YY-MM-DD).
LADATE	Date of the last file access (YY-MM-DD).
RDPASS	Read password required.
WRPASS	Write password required.
EXPASS	Execute password required.
VERSION	Version number of the original file.

It is set as follows when the file is opened:

- In OUTPUT or OUTIN mode it is set to 1.
- In INOUT or EXTEND mode it is set to the value of "BACKUP#" + 1 (where 255 + 1 = 0).

BACKUP# The number of the last file version saved. This informs the user whether or not the last version of the file has already been saved by the file saving system ARCHIVE.

If BACKUP# = VERSION, the newest version of the file has already been saved at least once.
If BACKUP# ≠ VERSION, the newest version of the file has not yet been saved.
BACKUP is set to zero if the file is opened in OUTPUT or OUTIN mode.

LARGE This attribute specifies whether the file is saved in its entirety during save runs using the file saving system ARCHIVE (i.e. LARGE=NO) or whether only those PAM blocks are saved which have been modified since the last save run (i.e. LARGE=YES).

BACKUP Save level when saved with the file saving system ARCHIVE.

DESTROY Data is destroyed when deleted.

AUDIT Monitoring by system exit routines.

In addition, the current storage allocation, distributed over private and public volumes, and the associated number of files are listed.

For a description of the 2nd line see STANDARD above.

T[RAITS] This operand specifies that the following file and volume information will be produced:

BLKTYPE	Buffer type (block type)
BLKSIZE	Buffer size (block size)
RECFORM	Record format
RECSIZE	Record size
KEYLEN	Key length for ISAM records
KEYPOS	Position of key for ISAM records
LOGLEN	Length of logical flag (ISAM only)
VALLEN	Length of value flag (ISAM only)
VALPROP	Continuation of value flag (ISAM only)
VSN/DEV/EXT	Volume serial number/device type/extents * No extent on this volume /nnn Number of extents on this volume
EXTCNT	Total number of extents; maximum number of extents: 319

The following information is issued for file generation groups:

GEN	Maximum number of generations allowed
BASE	Base value
LASTGN	Last generation
FIRSTGN	First generation
DISP	Value specified in the DISP operand of the CATALOG command.

For a description of the second line see STANDARD above.

ALL

This operand requests output of all information concerning the operands STANDARD, CATALOG and TRAITS.

P[ASSWORD]

This operand specifies whether the file is password-protected. The passwords themselves are not output. Users who have forgotten their passwords must obtain assistance from the system administrator.

For a description of the 2nd line see STANDARD above.

R[ESERVED]

This operand specifies that the total number of reserved PAM blocks is to be output, divided up according to whether they are on public or on private volumes.
The storage allocation specified by this operand also includes the PAM blocks occupied by the file generation groups.
This operand assumes the entry GEN=YES.
The operands STANDARD, CATALOG, TRAITS and PASSWORD are ignored.

A[CCESS]=R[EAD]

This operand lists files for which only read access is specified.

=W[RITE]

This operand lists files for which write access is specified.

If the ACCESS operand is not specified, catalog entry selection is not based on specific access methods.

B[ACKUP]=A
=B
=C
=D
=E

This operand requests all files with the specified save level. Several save levels may be specified, in which case all the files with any of the named save levels in the catalog entry will be listed.

If the BACKUP operand is omitted, catalog entry selection is not based on specific save levels.

CR[DATE]=date

This operand requests all the files with the specified creation date.

=(date,)

This operand requests all the files created from the specified date to the current date (inclusive).

=(,date)

This operand requests all the files with creation dates ranging from 19000101 to the date specified (inclusive).

=(date,date)
This operand requests all the files with creation dates lying in the specified range.

Date entries are specified in the form "YYMMDD" and must correspond to exact dates. The specifications Y[ESTERDAY], T[ODAY] and TOM[ORROW] are also accepted.

If the CRDATE operand is omitted, catalog entry selection is not based on specific creation dates.

EX[DATE]=date This operand requests all the files with the specified expiration date.

=(date,) This operand requests all the files with expiration dates ranging from the specified date to 1999-12-31 (inclusive).

=(,date) This operand requests all the files with expiration dates ranging from 19000101 to the specified date (inclusive).

=(date,date)
This operand requests all the files with expiration dates lying within the specified range.

Date entries are made in the form "YYMMDD" and must each correspond to an exact date. The entries Y[ESTERDAY], T[ODAY] and TOM[ORROW] are also accepted.

If the EXDATE operand is omitted, catalog entry selection is not based on specific expiration dates.

Format of date specifications:

For the selection criteria CRDATE, EXDATE and LADATE the date can be specified both as an absolute and as a relative value.

Absolute value
This refers to a specific date in the form YYYY-MM-DD, where YYYY is the year, MM the month and DD the day. When specifying the year, the century can be omitted, as can leading zeros for the specification of the month and the day.

Relative value
The date can also be specified as the distance in days from the current date, "-N" indicating the past, "+N" the future. Leading zeros can be omitted, but the signs are mandatory.
A simple method of specifying "immediate" dates is provided by the predefined terms Y[ESTERDAY] (specified as -1), T[ODAY] (specified as +0 or -0), and TOM[ORROW] (specified as +1).

EXT[ENTS]=value This operand outputs each disk file with exactly the requested number of extents.

=(value,)
This operand outputs all the disk files whose number of extents is greater than or equal to the value specified.

=(,value)
This operand outputs all the disk files whose number of extents is less than or equal to the value specified.

=(value,value)
This operand outputs all the disk files whose number of extents lies within the range specified.

For "value" all integers in the following range are permissible:

$$0 \leq \text{value} \leq 65535$$

If the EXTENTS operand is omitted, catalog entry selection is not based on the number of extents.

FCB[TYPE]=P[AM]
=S[AM]
=I[SAM]
=N[ONE]
This operand requests all the files having the specified access method. More than one access method can be specified, in which case all the files with one of the specified access methods are listed.
If the FCBTYPE operand is omitted, catalog entry selection is not based on specific access methods.

F[REE]SIZE=SIZE All the disk files for which not all the reserved lists are actually occupied are output.

=value Each disk file with exactly the requested number of reserved but not occupied PAM blocks is output.

=(value,)
All the disk files whose number of reserved but not occupied PAM blocks is greater than or equal to the value specified are output.

=(,value)
All the disk files whose number of reserved but not occupied PAM blocks is less than or equal to the value specified are output.

=(value,value)
All the disk files whose number of reserved but not occupied PAM blocks lies within the range specified are output.

For "value" all integers in the following range are permitted:

$$0 \leq \text{value} \leq 16777215$$

If the FREESIZE operand is omitted, catalog entry selection is not based on the number of PAM blocks which are not occupied.

FROM=CAT[ALOG] The information on the file(s) is read from the system catalog with the specified "catid". If no catalog ID is specified, the user's standard catalog ID is assumed.

=(vsn,device)
The information on the file(s) is read from the directory of the private disk with the specified vsn. The device type of the disk must be specified, the following entries for "device" being possible: D3455, D3465, D3468, D3470, D3475, D3480, D348E.
The operands VOLUME=, SUPPORT= and VTOC are prohibited if this operand is specified.
The "catid" specification is ignored if the FROM operand is specified.

LA[DATE]=date This operand requests all the files with the specified last access date.

=(date,) This operand requests all the files which were last accessed between the date specified and the current date (inclusive).

=(,date) This operand requests all the files whose last access date lies between 19000101 and the date specified (inclusive).

=(date,date)
This operand requests all the files whose last access date lies in the specified range.
Date entries are specified in the form "YYMMDD" and must correspond to exact dates. The specifications Y[ESTERDAY], T[ODAY] and TOM[ORROW] are also accepted.
If the LADATE operand is omitted, catalog entry selection is not based on specific access dates.

P[ASS]=R[DPASS]
=W[RPASS]
=E[XPASS]
=N[ONE]
This operand requests all the files that are protected by an appropriate password type. More than one password type can be specified. Any file identified by at least one of the password types specified is listed.

Meaning of password types:

RDPASS	File is read-protected
WRPASS	File is write-protected
EXPASS	File is execute-protected
NONE	File is not password-protected

If the PASS operand is omitted, catalog entry selection is not based on specific passwords.

L[IST]=target The requested information is output in table form to the specified output medium. The following specifications may be made for "target":

(SYSLST)	System file SYSLST
(SL)	System file SYSLST
(PRINT)	Printer
(PR)	Printer
Listfile	File

=(target,type)

The requested information is output in table form to the specified output medium, in which case the output format is defined. Possible specifications for "type" are:

STANDARD	Table specification (standard)
STD	Table specification (standard)
F[ILENAM]	File name only

Standard output is in the form of an edited list containing control characters.
When the operand LIST=(target,FILE) is used to specify the file names, the file names are issued separately, each record beginning with a blank (X'40').

When LIST is specified, the operands S, T, P, C and R are not permitted.
If the LIST operand is omitted, the requested information is output via SYSOUT (default value).

SAVE=Y[ES] This operand lists all the files whose last version has been changed at least once with the file saving system ARCHIVE; i.e. BACKUP# = VERSION.

=N[O] This operand lists all the files whose current version has not yet been saved with the file saving system ARCHIVE; i.e. BACKUP# ≠ VERSION.

If the SAVE operand is omitted, catalog entry selection is performed without regard to whether the file has been saved or not.

SHARE=Y[ES] This operand lists all the files which can also be accessed from other user IDs.

=N[O] This operand lists all the files to which only the owner has access.

If the SHARE operand is not specified, catalog entry selection is not based on access authorization.
However, if the user's own "userid" is not specified, only shareables files for which access authorization has been given are selected.

SIZE=F[REE]SIZE All those files are output which contain reserved pages that are not occupied.

=value Each disk file with the requested number of PAM blocks is listed. For values that are not multiples of three the preceding or subsequent multiple of three is assumed as the value.

=(value,) This operand outputs all the disk files whose number of PAM blocks is greater than or equal to the specified value.

=(,value) This operand outputs all the disk files whose number of PAM blocks is less than or equal to the specified value.

=(value,value)

This operand outputs all the disk files whose number of PAM blocks lies within the specified range.

For "value" all integers in the following range are permitted:

$$0 \leq value \leq 16777215$$

If the SIZE operand is omitted, catalog entry selection is not based on the number of PAM blocks allocated.

SORT=F[ILE]NAME — Files are output in alphabetical order (default value).

=N[O] — Files are output in the order in which they are listed in the catalog.

STATE=NOCLOS — This operand requests information concerning currently opened output files (and file generations) which were:

- opened normally with "OUT" (OPEN mode OUTIN, INOUT or OUTPUT)
- not closed in a previous session
- not closed in the current session because the job was terminated before completion (abnormal job termination - "pended indefinitely").

=PCLOSE — This operand requests information on files for which a dummy close was performed.

If this entry is omitted, the catalog entry selection is performed without regard to whether an output file is open or closed.

SUP[PORT]=
=PUB[LIC]
=PRD[ISC]

Catalog entries of files which include an entry for the type of volume specified are output. When both types of volumes are specified, the catlog entries of files on both public and private volumes are listed.

Where:

PUBLIC	Public volume
PRDISC	Private disk

If the SUPPORT operand is omitted, catalog entry selection is not based on specific volume types.

TYPE=FGG — This operand specifies that only information concerning file generation groups is to be output.

If this entry is omitted, the selection of catalog entries is unrestricted.

GEN=Y[ES] This operand requests information on all generations of a file generation group and is ignored if "filename" is not the name of a file generation group.

=[N]O No information concerning file generations is output (default value).

Operands		Types of catalog entry which may be output:		
TYPE =FGG	GEN =YES	File generation groups	File generations	Files
X	X	X	X	-
X	-	X	-	-
-	X	X	X	X
-	-	X	-	X

VOL[UME]=vsn This operand outputs all file names and status information for files which have an entry in the catalog for the specified volume.

VTOC=Y[ES] This operand outputs the VTOC catalog entries (from the F1 label of a private disk) on the basis of the last current status in the entire computer network.
If the specified file no longer resides on the private volume specified in the catalog entry, the catalog entry is deleted.
This entry causes all other keyword operands to be ignored.

=N[O] This operand outputs the TSOS catalog entries - on the basis of the last current status in an individual co-user. (Co-user: CPU for which a random access device was defined as shareable.)

Notes:

- Auxiliary function of the FSTATUS command:

  In interactive mode only; this command is ineffective in batch mode.

Operation	Operands
{FSTATUS / FSTAT}	,H[ELP]

HELP The syntax of the FSTATUS command is output.

If other operands are specified at the same time, they are ineffective.

- If the FSTATUS command is entered without operands, information on storage reservation and names of files under the user's own user ID (in alphabetical order) are output.

- Up to five positional operands can be specified.

- Any number of the operands CATALOG, STANDARD, TRAITS, PASSWORD and ALL can be specified in any order.

- In interactive mode, the BREAK function can be used to interrupt the output of FSTATUS information.

- If the operand VTOC=YES is specified, the volume must be assigned for the executing processor.

- If VTOC=YES is entered, the other specified key operands are ignored.

- The VTOC operand is ignored unless the file concerned is a private random access file. It is also rejected if the file name is only partially qualified or if the operand "GEN=YES" was specified.

- If a TSOSCAT entry exists, but no corresponding VTOC entry, the TSOSCAT entry is deleted, provided VTOC=YES has been specified.

- Specification of VTOC=YES causes the VTOC entry for the private volume to be read; this replaces the corresponding TSOSCAT entry. This action restores consistency between the VTOC and TSOS catalog entries, should this have been lost due to a file having been updated by another co-user.

Network catalog management (see also the "MSCF Multiprocessor system" manual)

Any "catid" or "userid" can be specified. A distinction is drawn between the following different cases:

- Without operands:
  All file entries from the user's own catalog belonging to the job-specific standard user ID are output.

- There is a catalog ID, but no user ID:
  All file entries from the catalog with the specified catalog ID belonging to the job-specific standard user ID are output. In this case, the command has the following format:
  FSTAT :catid:

- Only the user ID is specified:
  The file entries from the standard catalog belonging to the specified user ID are output.

Examples:

All files under the user's own user ID are output in alphabetical order:

```
/FSTAT
0000030 :V:$PM211023.BEISPIEL
0000846 :V:$PM211023.DVS.BAND
0000006 :V:$PM211023.DVS.KORR
0000012 :V:$PM211023.KOM.S-R-A
0000006 :V:$PM211023.PRI
0000009 :V:$PM211023.SCR.KOM
0000009 :V:$PM211023.TEST
:V:  PUBLIC:      7 FILES. RES=      918, FREE=      35, REL=      30 PAGES
```

All shareable files under the user ID $PM211 are output:

```
/FSTAT $PM211.
0000003 :V:$PM211.HELD.LADMOD
0000003 :V:$PM211.NEU.LINK.MSGID.PROG
0000015 :V:$PM211.NEU.LMS.PROG.MAKROS
0000003 :V:$PM211.NEU.QK
0000003 :V:$PM211.ZE.ANSCHR
0000003 :V:$PM211.ZE.DR.INFO
0000114 :V:$PM211.ZELL.NDFILE
:V:  PUBLIC:      7 FILES. RES=      144, FREE=       9, REL=       3 PAGES
```

In the case of files on private disk, an asterisk ("*") appears before the file name.

```
/FSTAT PRIVAT,ALL
0000087*:V:$PM211034.PRIVAT
  FCBTYPE = ISAM     VSNTYPE = PVT       LASTPG  = 0000086   2ND ALLO= 00009
  SHARE   = NO       ACCESS  = WRITE
  ACCESS# = 001      CRDATE  = 85-03-20  EXDATE  = 85-03-20  LADATE  = 85-03-20
  RDPASS  = NONE     WRPASS  = NONE      EXPASS  = NONE
  VERSION = 001      BACKUP# = 000       LARGE   = NO        BACKUP  = A
  DESTROY = NO       AUDIT   = NONE
  BLKTYPE = STD      BLKSIZE = 002048    RECFORM = (V,N)     RECSIZE = 00000
  KEYLEN  = 008      KEYPOS  = 00005
  VSN/DEV/EXT =      C0016M/D3468/002
  EXTCNT  =   2
:V:  PRIVATE:     1 FILE.  RES=       87, FREE=       1, REL=       0 PAGES
```

All files whose names begin with DVS. are output:

```
/FSTAT DVS.
0000066 :V:$PM211034.DVS.BEISP.1
0000006 :V:$PM211034.DVS.PL.BILB
0001074 :V:$PM211034.DVS.PLATTE
0000006 :V:$PM211034.DVS.TABELLE.2-2
0000006 :V:$PM211034.DVS.VERTEILER
:V:  PUBLIC:      5 FILES. RES=     1158, FREE=       7, REL=       0 PAGES
```

The number of PAM blocks used is output:

```
/FSTAT ,R
:V:  PUBLIC:     45 FILES. RES=     1746, FREE=     178, REL=     135 PAGES
:V:  PRIVATE:     5 FILES. RES=     1248, FREE=      34, REL=      30 PAGES
```

The number of PAM blocks used by all pubsets under the user ID (in this case V and Z) is output.
The specifications in the totals line refer to the two pubsets.

```
FSTAT :*:,R
:V:  PUBLIC:      45 FILES. RES=     1746, FREE=      178, REL=      135 PAGES
:V:  PRIVATE:      5 FILES. RES=     1248, FREE=       34, REL=       30 PAGES
:Z:  PUBLIC:      19 FILES. RES=      117, FREE=       27, REL=        3 PAGES
SUM  PUBLIC:      64 FILES. RES=     1863, FREE=      205, REL=      138 PAGES
SUM  PRIVATE:      5 FILES. RES=     1248, FREE=       34, REL=       30 PAGES
```

Specific information is requested for selected files:

```
/FSTAT DATEI,ALL
0000087 :V:$PM211034.DATEI
  FCBTYPE = ISAM     VSNTYPE = PUB       LASTPG  = 0000086   2ND ALLO= 00012
  SHARE   = YES      ACCESS  = WRITE
  ACCESS# = 018      CRDATE  = 85-03-19  EXDATE  = 85-03-19  LADATE  = 85-03-20
  RDPASS  = YES      WRPASS  = YES       EXPASS  = YES
  VERSION = 001      BACKUP# = 001       LARGE   = NO        BACKUP  = A
  DESTROY = NO       AUDIT   = NONE
  BLKTYPE = STD      BLKSIZE = 002048    RECFORM = (V,N)     RECSIZE = 00000
  KEYLEN  = 008      KEYPOS  = 00005
  VSN/DEV/EXT =      PUBV05/D3475/002
  EXTCNT  =   2
:V:  PUBLIC:       1 FILE.  RES=       87, FREE=        1, REL=        0 PAGES

/FSTAT ASS.PRO,STANDARD
0000003 :V:$PM211034.ASS.PRO
  FCBTYPE = ISAM     VSNTYPE = PUB       LASTPG  = 0000002   2ND ALLO= 00009
:V:  PUBLIC:       1 FILE.  RES=        3, FREE=        1, REL=        0 PAGES

/FSTAT ASS.PRO,CATALOG
0000003 :V:$PM211034.ASS.PRO
  SHARE   = NO       ACCESS  = WRITE
  ACCESS# = 022      CRDATE  = 85-02-26  EXDATE  = 85-02-26  LADATE  = 85-03-08
  RDPASS  = NONE     WRPASS  = NONE      EXPASS  = NONE
  VERSION = 001      BACKUP# = 001       LARGE   = NO        BACKUP  = A
  DESTROY = NO       AUDIT   = NONE
:V:  PUBLIC:       1 FILE.  RES=        3, FREE=        1, REL=        0 PAGES

/FSTAT BEISPIEL,TRAITS
0000030 :V:$PM211034.BEISPIEL
  BLKTYPE = STD      BLKSIZE = 002048    RECFORM = (V,N)     RECSIZE = 00000
  VSN/DEV/EXT =      PUBV11/D3475/001
  EXTCNT  =   1
:V:  PUBLIC:       1 FILE.  RES=       30, FREE=       30, REL=       30 PAGES

/FSTAT DATEI,PASSWORD
0000087 :V:$PM211034.DATEI
  RDPASS  = YES      WRPASS  = YES       EXPASS  = YES
:V:  PUBLIC:       1 FILE.  RES=       87, FREE=        1, REL=        0 PAGES
```

The names of opened output files are requested:

```
/FSTAT ,STATE=NOCLOS
0000030 :V:$PM211034.BEISPIEL
:V:  PUBLIC:       1 FILE.  RES=       30, FREE=       30, REL=       30 PAGES
```

The names of all file generation groups are output:

```
/FSTAT TYPE=FGG
0000000 :V:$PM211034.C.DGG (FGG)
0000000 :V:$PM211034.DGG1 (FGG)
0000000 :V:$PM211034.DGG2 (FGG)
:V:  PUBLIC:      3 FILES. RES=        0, FREE=        0, REL=        0 PAGES
```

The names of all file generation groups, including file generations, are output:

```
/FSTAT TYPE=FGG,GEN=YES
0000000 :V:$PM211023.DGG1 (FGG)
0000006 :V:$PM211023.DGG1(*0001)
0000009 :V:$PM211023.DGG1(*0002)
0000012 :V:$PM211023.DGG1(*0003)
0000012 :V:$PM211023.DGG1(*0004)
0000006 :V:$PM211023.DGG1(*0005)
:V:  PUBLIC:      6 FILES. RES=       45, FREE=        4, REL=        0 PAGES
```

All files, including file generations under the user's own user ID, are output:

```
/FSTAT ,GEN=YES
0000030 :V:$PM211023.BEISPIEL
0000000 :V:$PM211023.DGG1 (FGG)
0000006 :V:$PM211023.DGG1(*0001)
0000009 :V:$PM211023.DGG1(*0002)
0000012 :V:$PM211023.DGG1(*0003)
0000012 :V:$PM211023.DGG1(*0004)
0000006 :V:$PM211023.DGG1(*0005)
0000846 :V:$PM211023.DVS.BAND
0000006 :V:$PM211023.DVS.KORR
0000012 :V:$PM211023.KOM.S-R-A
0000006 :V:$PM211023.PRI
0000009 :V:$PM211023.SCR.KOM
0000009 :V:$PM211023.TEST
:V:  PUBLIC:     13 FILES. RES=      963, FREE=       39, REL=       30 PAGES
```

All files under the user ID are output (corresponds to FSTAT without operands).

```
/FSTAT *
0000030 :V:$PM211023.BEISPIEL
0000000 :V:$PM211023.DGG1 (FGG)
0000846 :V:$PM211023.DVS.BAND
0000006 :V:$PM211023.DVS.KORR
0000012 :V:$PM211023.KOM.S-R-A
0000006 :V:$PM211023.PRI
0000009 :V:$PM211023.SCR.KOM
0000009 :V:$PM211023.TEST
:V:  PUBLIC:      8 FILES. RES=      918, FREE=       35, REL=       30 PAGES
```

All files whose names consist of only three characters are listed.

```
/FSTAT ///
0000003 :V:$PM211034.ASS
0000030 :V:$PM211034.DAT
0000003 :V:$PM211034.M.1
0000009 :V:$PM211034.M.2
0000009 :V:$PM211034.M.3
:V:  PUBLIC:      5 FILES. RES=       54, FREE=       28, REL=       24 PAGES
```

All files with names beginning with the letter D or a letter in the range from X to Z (inclusive) are listed.

```
/FSTAT <D,X:Z>*
0000030 :V:$PM211034.DAT
0000087 :V:$PM211034.DATEI
0000087 :V:$PM211034.DATEI.KOPIE
0000000 :V:$PM211034.DGG1 (FGG)
0000000 :V:$PM211034.DGG2 (FGG)
0000003 :V:$PM211034.DO.BEISPIEL
0000066 :V:$PM211034.DVS.BEISP.1
0000006 :V:$PM211034.DVS.PL.BILB
0001074 :V:$PM211034.DVS.PLATTE
0000006 :V:$PM211034.DVS.TABELLE.2-2
0000006 :V:$PM211034.DVS.VERTEILER
0000003 :V:$PM211034.ZE.ASS
:V:  PUBLIC:     12 FILES. RES=     1368, FREE=      16, REL=      3 PAGES
```

All those files are listed whose names are not 4 characters in length and which are read-protected/write-protected.

```
/FSTAT -////,PASS=(R,W)
0000087 :V:$PM211034.DATEI
0000087 :V:$PM211034.DATEI.KOPIE
```

All files on private disk C0016M are output to SYSLST in table form.

```
/FSTAT ,VOLUME=C0016M,L=(SL)
```

FILE STATUS ,VOLUME=C0016M,L=(SL) 07

FILENAME	PAM PAGES	FREE PAGES	SECOND ALLOC.	FCB TYPE	SHARE	ACCESS	PASS-WORDS	BKL	#EXT	VOLUME
*:V:$PM211034.DO	3	1	9	ISAM	NO	WRITE		A	1	C[illegible]M
*:V:$PM211034.FROH	30	30	9	SAM	NO	WRITE		A	2	C[illegible]M
*:V:$PM211034.PRIVAT	87	1	9	ISAM	NO	WRITE		A	2	C0016M
*:V:$PM211034.SAVE	1077	1	9	ISAM	NO	WRITE		A	2	C0016M
*:V:$PM211034.SYSTEM	51	1	9	ISAM	NO	WRITE		A	2	C0016M
PRIVATE SPACE: 5 FILES 1248										

In the PAM PAGES column, the entire memory space occupied by the volumes in use is output. If a file reserves space on more than one volume, the vsn is preceded by an asterisk ("*") in the VOLUME column.

The temporary file #WORK is output ("#" is the symbol for temporary file). The file name is specified via the internal file name S.TMP.nnnn.file, where "nnnn" stands for the tsn.

```
/FSTAT #WORK,ALL
0000087 :V:$PM211034.S.TMP.8166.WORK
  FCBTYPE = ISAM     VSNTYPE = PUB       LASTPG  = 0000086   2ND ALLO= 00012
  SHARE   = NO       ACCESS  = WRITE
  ACCESS# = 001      CRDATE  = 85-03-20  EXDATE  = 85-03-20  LADATE  = 85-03-20
  RDPASS  = NONE     WRPASS  = NONE      EXPASS  = NONE
  VERSION = 001      BACKUP# = 000       LARGE   = NO        BACKUP  = E
  DESTROY = NO       AUDIT   = NONE
  BLKTYPE = STD      BLKSIZE = 002048    RECFORM = (V,N)     RECSIZE = 00000
  KEYLEN  = 008      KEYPOS  = 00005
  VSN/DEV/EXT =      PUBV09/D3475/001
  EXTCNT  =   1
:V:  PUBLIC:      1 FILE.  RES=       87, FREE=        1, REL=        0 PAGES
```

All temporary files are output (temporary-file symbol = "#"). The temporary files are output using the file name S.TMP.nnnn.file (where "nnnn" stands for the tsn).

```
/FSTAT #
0000087 :V:$PM211034.S.TMP.8166.TEIL
0000087 :V:$PM211034.S.TMP.8166.TEMP
0000087 :V:$PM211034.S.TMP.8166.WORK
:V:  PUBLIC:      3 FILES. RES=      261, FREE=        3, REL=        0 PAGES
```

HOLD HOLD TFT ENTRY

The HOLD command is used to place an entry in the Task File Table (TFT) in the HOLD status. This serves to defer a subsequent RELEASE command (or REL macro) for this TFT entry until a DROP command with the relevant file link name is issued.

Operation	Operands
HOLD	[link]

link — This operand specifies the file link name of a TFT entry to be placed in the HOLD status. If no file with this name existed previously, a new TFT entry is created for this link name (max. 8 bytes). Further entries can be made in this TFT entry by means of a subsequent FILE command. If "link" is not specified, the first TFT entry with the link name C'␣␣␣␣␣␣␣␣' is processed.

Example 1:

/FILE LINK=X,...	An entry with the name X is created in the TFT.
/HOLD X	The HOLD status indicator is set for TFT entry X.
/RELEASE X . .	The entry X is not deleted, as it is in the HOLD status. However, the command (or REL macro) sets the RELEASE indicator in the TFT entry X.
/DROP X	Only now does the RELEASE command (or REL macro) take effect, the TFT entry X being deleted, and any associated private devices released.

Example 2:

/HOLD Y	A TFT entry with the name Y is created, and its HOLD status indicator set.
/FILE LINK=Y,... . . .	This command uses the existing TFT entry Y, and makes entries in it.
/DROP Y .	The HOLD status for entry Y is canceled. As no RELEASE command has been issued, no operation is performed.
/RELEASE Y	The entry Y is no longer subject to HOLD. The RELEASE command thus has the immediate effect of deleting entry Y.

Example 3:

```
/FILE LINK=A
/HOLD A
```

/CHANGE A,B	The name of TFT entry A is changed to B, but everything else in the TFT entry remains unchanged, e.g. entry B is now subject to the HOLD status.
/DROP A	This command generates an error message as there is no longer an entry with this link name.
/RELEASE B	The RELEASE indicator is set in TFT entry B, but the command is not yet executed as HOLD is in force.
/DROP B	RELEASE becomes effective, i.e. entry B (previously entry A) is deleted.

IMPORT CREATE CATALOG ENTRY FOR PRIVATE FILES

This command catalogs files contained on a private disk. In so doing, it transfers the entries from the disk's F1 label to the system catalog (volume import).

Only those files can be cataloged which were set up on the private disk under the same user ID as that under which the calling task is executed.

This command is particularly suited for the rapid introduction of multiple files from a private disk into the system.

When importing file generation groups whose generations are located on different disks, certain rules must be observed; these are listed in the notes at the end of the format description.

A function with the opposite effect to that of the IMPORT command is contained in the ERASE command: operand VOLUME=vsn (volume export).

Operation	Operands
IMPORT	[pathname,]VOLUME=vsn,DEVICE=device [,REPLACE={YES | ABS | NO}] [,LIST={{YES | NO | ONLY} | ({YES | NO | ONLY},{SYSLST | SYSOUT | BOTH})}] [,GEN={YES | NO}] [,PVSID=catid]

"pathname" stands for: [$userid.]filename

userid	This operand denotes the user ID associated with the file. If no entry is made, the user's own user ID is assumed. Only the user's own user ID is permitted.
filename	This operand specifies a fully or partially qualified file name or the name of a file generation group. If this operand is omitted, the DMS catalogs all the files belonging to the calling user.
VOLUME=vsn	This operand specifies the volume serial number of the disk (6 characters).

DEVICE=device — The type of device on which the private disk is to be mounted, e.g. D3465 (see also the FILE command, operand DEVICE=).

REPLACE=YES — Entries from the F1 label replace entries with the same file name in the system catalog.
The following situations may occur:

1. The file whose catalog entry already exists in the system catalog is located on public volumes. If permitted by the protection attributes of the file, the file is deleted. Otherwise nothing happens.

2. The file whose catalog entry already exists in the system catalog is located on private disks, but does not begin on the disk specified in the IMPORT command. If permitted by the protection attributes of the file, the catalog entry for the file is deleted (ERASE CATALOG). Otherwise nothing happens.

3. The file whose catalog entry already exists in the system catalog is located on private disks and begins on the disk specified in the IMPORT command. In this case nothing happens. The DMS regards the entry in the system catalog as a valid description of the file to be imported.

=ABS — Entries from the F1 label replace entries with the same file name in the system catalog, provided that the catalog entries have the same vsn as the disk to be imported.

Three different types of situations can occur:

1. As with REPLACE=YES

2. As with REPLACE=NO

3. The catalog entry of the file which is to be imported already exists in the system catalog. The file is on private disk and begins on the disk which was specified in the IMPORT command. In this case the catalog entry is deleted and replaced by the current entry (return code 8).
If this is not possible due to a file lock, nothing happens, since an import is in this case unnecessary (return code 9).

=NO — The DMS creates entries in the system catalog only for those files which are not yet cataloged.

LIST=YES
=NO — This operand causes or suppresses output of a list of return information concerning the processing of files.

=ONLY — This operand outputs only a list of files located on the private disk; no import operation is performed. The contents of the list depend on the operand "filename"; special attention should be paid to the following:

- The information supplied for each file explains how these files would have been processed if they had been imported into the system.

- If an entry for a file on the private disk already exists in the system catalog, the DMS does not check the protection attributes for this file. Should the user wish to import the file at some later time, and if it is necessary to delete the previously cataloged file in order to to this, the user himself must ensure that this file can be deleted (ACCESS=WRITE, retention period expired, passwords specified).

={SYSOUT / SYSLST / BOTH}

These keywords denote the symbolic system file to which the list is to be output (SYSOUT - terminal; SYSLST - printer; BOTH - both terminal and printer).

In the SYSOUT listing, the information appears only in the form of a code, while the SYSLST listing outputs the corresponding message text.

The following messages can be output:

Code on SYSOUT	Message on SYSLST	Meaning
0	FILE DID NOT EXIST	Processing successful. No file with this name existed previously.
1	FILE HAS BEEN ERASED	A file of the same name already existed and has been overwritten.
2	FILE EXISTS AND REPLACE=NO	A file of the same name already existed and has not been overwritten.
3	FILE IS PROTECTED (ERASE ERROR or FILE IS IN USE)	A file of the same name already exists and could not be erased because of active protection functions (ACCESS= READ, WRPASS=, etc.), or the file to be imported is currently being processed (FILE IS IN USE).
4	ERROR DURING $RDCT (IN CATALOG)	Unrecoverable error (CATALOG MANAGEMENT).
5	FILE ALREADY ON PRIVATE	The file is already cataloged as a foreign file on a private disk and REPLACE=ABS is not specified.
6	$SCAN ERROR (ON VTOC)	Error in command processing.
7	NOT ALLOWED TO RE-CATALOG A GENERATION	Import of a file generation is not permissible.
8	C.E. HAS BEEN REPLACED	The catalog entry for the disk already existed and has been overwritten.
9	FILE IS IN USE	The catalog entry for the disk already exists and the file is locked.

GEN=YES — This operand imports for a file generation group the group entry and all the generations that begin on the private disk. If the private disk contains only generations of a file generation group but not the group entry, then these generations will be cataloged only if the group entry is already cataloged.

=NO — This operand imports only the group entry of a file generation group.

PVSID=catid — This operand specifies the catalog which is to contain the catalog entries.
If this operand is not specified, the standard catalog ID of the user is assumed.

Notes:

The functions of the IMPORT command and the ERASE CATALOG, VOLUME=vsn command are not exact opposites:

When a volume is exported, the DMS deletes the catalog entries of all the files occupying space on this volume.

If the same volume is imported again, the DMS creates catalog entries only for those files that begin on this volume (i.e. files that received space on this disk in the primary allocation).

When importing or exporting private disks on which file generation groups are located, the following points should be borne in mind:

1. An IMPORT command catalogs only those generations whose group entry is contained either on the disk specified or in the system catalog.

   For a file generation group that is distributed over several disks and is not yet cataloged, this has the following effect:

   If the disk that does not contain the group entry is imported first, followed by the disk with the group entry, the catalog entries for the generations contained on the first disk will be missing.
   This can be remedied by issuing either a new IMPORT command for each volume affected, or a FILE command (operand STATE=FOREIGN) for each uncataloged generation.

Example:

Two file generations are distributed over three private disks, as follows:

(GRP stands for group entry)

The generations and group entries named are not contained in the system catalog.

The following command sequence is issued:

```
/IMPORT VOLUME=DSK001,DEVICE=D3465
/IMPORT VOLUME=DSK002,DEVICE=D3465
/IMPORT VOLUME=DSK003,DEVICE=D3465
```

After these commands have been executed, the following files and group entries are cataloged:

```
FGG1GRP  FGG2GRP
FGG1(*1) FGG2(*1)
FGG1(*2) FGG2(*3)
FGG1(*3)
```

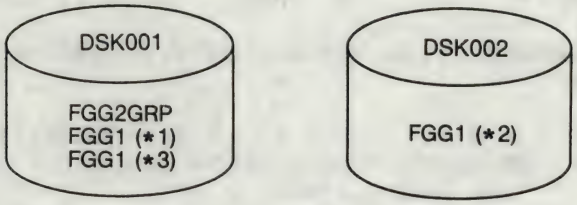
(It should be noted that there is no sequence of commands which would have cataloged all the generations.)

Generation FGG2(*2) may subsequently be imported, e.g. with a FILE command.

2. If a private disk contains only individual generations of a file generation group, but not the group entry, gaps may occur in the file generation group when this volume is exported.

Example:

One file generation group is distributed over two private disks as follows:



(GRP stands for group entry)

The command

/ERASE CATALOG,VOLUME=DSK002

deletes generation 2 from the system catalog, while generations 1 and 3 remain cataloged as before. This means that there is a gap in the file generation group.
However, this situation affects only the catalog entries; on the disks the file generation group is complete.

Further examples:

- /IMPORT ,VOLUME=G8002M,DEVICE=D3468,LIST=(YES,SYSOUT)

```
0 $KRZ.MAN.KDO
0 $KRZ.MAN.MK
0 $KRZ.MAN.MKS
```

All files under the user ID $KRZ that are located on the private disk are imported, as the return information output confirms (code=0).

- /IMPORT MAN.MAKRO,VOLUME=G8002M,DEVICE=D3468,LIST=(YES,BOTH)

```
0 $KRZ.MAN.MAKRO
```

The IMPORT command imports the file MAN.MAKRO. In addition to output to SYSOUT, there appears on SYSLST the corresponding return information for code 0:

```
* * *  I M P O R T   O U T P U T   L I S T   * * *  14:39:19  02/05/83
IMPORT MAN.MAKRO,VOLUME=G8002M,DEVICE=D3468,LIST=(YES,BOTH)

  OLD USER-ID CODE          NEW USER-ID CODE                FILE NAME
                          FILE DID NOT EXIST            $KRZ.MAN.MAKRO
```

- The file DUPLIKAT under the user ID $PM211023 is located on a private disk and is to be imported. However, a file with the same name already exists under this user ID on a public volume:

```
/FSTAT DUPLIKAT
0000003 :V:$PM211023.DUPLIKAT
:V:  PUBLIC:      1 FILE.  RES=          3, FREE=        1, REL=       0, PAGES

   /IMPORT DUPLIKAT,VOLUME=G8002M,DEVICE=D3468,LIST=(YES,BOTH)
      2 :V:$PM211023.DUPLIKAT

   * * *   I M P O R T   O U T P U T   L I S T   * * *  14:41:10  02/05/83
   IMPORT DUPLIKAT,VOLUME=G8002M,DEVICE=D3468,LIST=(YES,BOTH)
     OLD USER-ID CODE          NEW USER-ID CODE                 FILE NAME
                            FILE EXISTS/REPLACE=NO     :V:$PM211023.DUPLIKAT
```

- The file is not imported, as the SYSOUT and SYSLST listings show:

```
   /IMPORT DUPLIKAT,VOLUME=G8002M,DEVICE=D3468,LIST=(YES,BOTH),REPLACE=YES
      1 :V:$PM211023.DUPLIKAT

   * * *   I M P O R T   O U T P U T   L I S T   * * *  14:45:04  02/05/83
   IMPORT DUPLIKAT,VOLUME=G8002M,DEVICE=D3468,LIST=(YES,BOTH),REPLACE=YES
     OLD USER-ID CODE          NEW USER-ID CODE                 FILE NAME
                            FILE HAS BEEN ERASED       :V:$PM211023.DUPLIKAT
```

- The following SYSOUT listing names all the files contained on private disk under the user ID $KRZ. No file import takes place since LIST=ONLY was specified. Here the user can ascertain from the codes what would have happened in the event of an import:

```
   /IMPORT ,VOLUME=G8002M,DEVICE=D3468,LIST=(ONLY,SYSOUT)

      2 $KRZ.MAN.KDO
      0 $KRZ.MAN.KOM
      2 $KRZ.MAN.MAK
      0 $KRZ.MAN.MKR
```

The files MAN.KDO and MAN.MAK are imported only if the operand REPLACE=YES is specified and if the protection attributes for the like-named files permit it (cf. previous example).
The other two files can be imported without the necessity for additional entries in the IMPORT command which differ from the default values.

- 7 file generations (*1 through *7), whose group entry no longer exists (in the catalog and in the F1 label), are located on private disk. The generations (*2) and (*3) are to be imported.

```
/CAT C.DGG,GEN=5,FIRST=2,BASE=3,DEVICE=D3468,VOLUME=C0016M
/FSTAT C.DGG,GEN=YES,ALL
0000000*:V:$PM211023.C.DGG (FGG)
   SHARE    = NO       ACCESS   = WRITE
   ACCESS#  = 000      CRDATE   = 85-03-20  EXDATE   = 85-03-20  LADATE   = NONE
   RDPASS   = NONE     WRPASS   = NONE      EXPASS   = NONE
   VERSION  = 000      BACKUP#  = 000       LARGE    = NO        BACKUP   = A
   DESTROY  = NO       AUDIT    = NONE
   GEN      = 00005    BASE     = 00003     LASTGN   = 00003     FIRSTGN  = 00002
   DISP     = CYCLE
   VSN/DEV/EXT =       C0016M/D3468
:V:  PRIVATE:     1 FILE.  RES=        0, FREE=        0, REL=        0 PAGES
/IMPORT ,VOLUME=C0016M,DEVICE=D3468,GEN=YES
    OLD USER-ID CODE          NEW USER-ID CODE                 FILE NAME
         FILE EXISTS/REPLACE=NO     :V:$PM211023.C.DGG
         FILE EXISTS/REPLACE=NO     :V:$PM211023.DUPLIKAT
/FSTAT C.DGG,GEN=YES
0000000*:V:$PM211023.C.DGG (FGG)
0000003*:V:$PM211023.C.DGG (*0002)
0000003*:V:$PM211023.C.DGG (*0003)
:V:  PRIVATE:     3 FILE.  RES=        6, FREE=        2, REL=        0 PAGES
```

The group entry must be newly created with the CATALOG command for file generations without a group entry which are to be imported:
C.DGG(*2) is the oldest generation to be imported (FIRST=2).
C.DGG(*3) is the newest generation to be imported (BASE=3).
The generation number (*3) is also the base value to which all the relative generation numbers relate. If the BASE operand is not specified, the system imports for BASE the value of the first operand; i.e. only one generation is imported (see also the CAT command, operand BASE).

PASSWORD SPECIFY PASSWORD

The PASSWORD command provides a task with passwords for file opening or for access to the catalog entry. For this purpose, a list of passwords (password table) is created which can be extended or reduced in stages, or also deleted, by means of subsequent PASSWORD commands.

This password table is searched during processing of the CATALOG, FILE and ERASE commands, and also upon file opening, if a password is needed for file access and this password is not contained in the File Control Block of the program. If the required password is found in the table, access is permitted.

At the end of the job the password table is deleted.

Operation	Operands
PASSWORD	[{password | (password,...)}][,REL=Y[ES]]

password — This operand specifies a password of up to 4 bytes in length; it is represented as follows:

- C'x' "x" consists of 1 to 4 alphanumeric characters and special characters;
- X'n' "n" consists of 1 to 8 hexadecimal digits;
- d "d" is a decimal integer consisting of 1 to 8 digits, whose value is converted to a binary value.

A password with the value X'00000000' is ignored. Up to 63 passwords can be specified in one PASSWORD command.

REL=Y[ES] — This operand specifies that the passwords defined in the PASSWORD command are to be deleted from the job's password table, so that full password protection is again established.

If no password is specified in conjunction with the REL operand, the job's entire password table is deleted.

Notes:

- The following applies to the use of passwords:

  If both a write password and a read password have been specified for a file, the write password assumes the function of both passwords, i.e. if the write password is entered, the user can both read and write to the file.

  If only a read password has been specified for a file, this must be entered for reading and/or writing. An existing read password must be entered when modifying the write password.

  If both a read password and an execute password have been specified for a file, the read password assumes the function of the execute password, i.e. entering the read password allows both reading and execution.

- The protection afforded by passwords can be enhanced by means of the internal encryption (encoding) of these passwords by the system. In memory dumps, for example, the passwords appear in coded form only. Password encryption is specified at system generation (PARAM ENCRYPT,Y).

- In addition, the protection afforded by passwords can be further enhanced by limiting the permissible number of invalid entry attempts.

Remote file access (see also the "RFA" manual)

- The PASSWORD command is passed on automatically to all RFA partner tasks by the requesting job.

- The passwords are not encrypted before transmission to the remote processor. Encryption takes place immediately afterwards, provided the computer uses password encryption.

Example:

In interactive mode, input includes the following commands:

```
/LOGON ...

/PASSWORD X'51EF' ———————————————————————————————————— 01)
/CAT M.NEU,STATE=U,RDPASS=C'OR' ———————————————————————— 02)
/PRINT M.NEU
%  SCP0860 FILE PROTECTED BY A READ PASSWORD. PRINT REQUEST REJECTED FOR
   :V:$PM211023.M.NEU.
/COPY M.NEU,M.NEU.KOPIE
%  DMS05F3 REQUIRED PASSWORD IS NOT IN PASSWORD TABLE. COMMAND TERMINATED
/PASSWORD C'OR' ——————————————————————————————————————— 03)
/COPY M.NEU,M.NEU.KOPIE
/PASSWORD REL=Y ——————————————————————————————————————— 04)
/ER M.NEU
%  DMS0801 ERASE ERROR ON FILE :V:$PM211023.M.NEU
%  DMS05BF REQUIRED PASSWORD IS NOT IN PASSWORD TABLE. COMMAND TERMINATED
/ER M.NEU.KOPIE ——————————————————————————————————————— 05)
/FSTAT M.NEU
0000006 :V:$PM211023.M.NEU
:V:  PUBLIC:      1 FILE.  RES=        6, FREE=        1, REL=        0 PAGES

/LOGOFF
```

01) The first PASSWORD command in this job serves to set up its password table and enter the password X'51EF' in it. Subsequently, access is permitted to all files protected in any way by this password. After entry, this command is always rendered illegible on the printer listing.

02) A read password for the file M.NEU is defined by means of the CATALOG command. The password is hereby entered in the catalog, but not recorded in the job's password table. The subsequent PRINT or COPY command thus leads to a corresponding error message.

03) The PASSWORD command enters the password C'OR' in the job's password table. The subsequent PRINT and COPY commands are thus performed as desired.

04) This PASSWORD command causes the password table to be deleted; the subsequent ERASE command is rejected.

05) The file M.NEU.KOPIE can be deleted because it is not password-protected.

RDTFT OBTAIN INFORMATION FROM TFT AND TST

By means of the RDTFT command, the user can output status information from the TFT relating to the files and devices currently in use, and also obtain information from the associated TST (see the FILE command).

Operation	Operands
RDTFT	[FILE=pathname] [,LINK={link / (BLANK,no)}] [,L[INKAGE]][,S[ECURITY]][,F[CB]][,V[OLUMES]] [,ALL]

FILE=pathname — The status information relating to the TFT entry associated with "filename" is output. Otherwise selection of the entry is not dependent on "filename".

"pathname" stands for: [:catid:][$userid.]filename

catid — This operand denotes the catalog identifier associated with the file. If this entry is omitted, the default catalog ID assigned to the user ID is assumed.

userid — This operand denotes the user identification associated with the file. If this operand is omitted, the user's own user ID is assumed.

filename — This operand specifies the partially or fully qualified file name (up to 41 characters). If the file name does not begin with a user ID, it is prefixed by the user ID of the job. If the file is a file generation of a group, the absolute value of the generation number must be specified. When temporary files are specified, the internal file name is output.

LINK=link — This operand specifies the file link name.

If the LINK operand is specified, only status information relating to the TFT entries associated with the file link name is output. Otherwise the selection of the entry is not dependent on the file link name.

=(BLANK,no) — This operand specifies the number of the TFT entry which has the link name C'␣␣␣␣␣␣␣␣'.

Such entries are generated by the OPEN macro if no link name or LINK=C'␣␣␣␣␣␣␣␣' is specified in the FCB (not recommended).

If the FILE and LINK operands are omitted, status information relating to all the TFT entries belonging to the job is output.

L[INKAGE]

If this operand is specified, the user receives linkage information in the following format:

```
STATUS  =ACTIVE/INACTIVE [HOLD] [DEV RELEASED]

                                        ⎡RELEASE        ⎤
COMMAND =FILE/OPEN     [DELAYED=        ⎨               ⎬]
                                        ⎣(RELEASE,KEEP) ⎦

DATA    =[DATA ON TAPE VOLUME COUNT]
```

where:

STATUS specifies the file status, the HOLD status, and whether the file was released after being closed.

COMMAND indicates whether the TFT entries were created by means of a FILE command/macro or an OPEN macro.

DELAYED indicates whether a HOLD command caused a RELEASE action to be delayed.

DATA specifies the number of tape devices that contain data.

S[ECURITY]

The user can use this operand to obtain information concerning file security and the retention period. Such information is output in the following format:

```
RET PERIOD=                    SECLEV=HIGH|LOW
OVWRITE PROTECT=[YES|NO]       BYPASS
```

where:

OVWRITE corresponds to the specifications in the OPR qualification (overwrite protection).
Cf. the BYPASS operand in the FILE command.

F[CB]

If the operand FCB is specified, the following file characteristics are output:

```
FCBTYPE=[access method]   OPEN =[open mode]
RECFORM=[record format]   RECSIZE=[record length]
BLKSIZE=[block size]      BUFOFF =[number|L]
KEYPOS =                  KEYLEN =
LOGLEN =                  VALLEN =
VALPROP=                  DUPEKY =
PAD    =                  OVERLAP=
SHARUPD=                  WROUT  =[YES/NONE]
LABEL  =[label type]      TPMARK =[YES/NO]
CODE   =[code type]       TRANS  =[YES/NONE]
FSEQ   =[UNK|NEW]         WRCHK  =[YES/NO]
```

V[OLUMES]

If this operand is specified, the device characteristics of the file are output in the following format:

```
DEVICE     =[device type]  TSET=[device name]
VSN/DEV    =VOL-SER-#/[device type code]
```

The device type code is output only if the device is loaded.

ALL This operand specifies that all the information described above is to be output for each TFT entry specified.

Note:

The TFT entries are output sorted according to their file link names.

Output format:

For files on public disk:

```
%                       LINK=linkname
                        FILE=filename
```

For files on private disk:

```
%DVOL DEV               LINK=linkname
                        FILE=filename
```

Remote file access (see also the "RFA" manual)

If the TFT entry for a file located in a remote system is to be displayed with the aid of this command, the format of the display field is as follows:

```
%R                      LINK=linkname
                        FILE=filename
```

"R" means that the file concerned is a remote file. The catalog containing the file is specified within the expression FILENAME by means of the "catid" entry.

Examples:

- STATUS information on the TFT entries associated with the file TRICK is requested:

```
/RDTFT FILE=TRICK
%          LINK=EDRPRIMR
           FILE=TRICK
%          LINK=IOOLINK
           FILE=TRICK
```

- STATUS information on the TFT entry associated with the file link name EDTISAM is requested, together with additional details on file security:

```
/RDTFT LINK=EDTISAM,SECURITY
%          LINK=EDTISAM
           FILE=TRICK
 RETENT. PERIOD =00015      SECLEV  =  HIGH
 OVWRITE PROTECT=NO
```

- Information on the HOLD status and the success of the RELEASE action is obtained using the LINKAGE operand:

```
/HOLD EDTISAM
/REL EDTISAM
/RDTFT LINK=EDTISAM,LINKAGE
%          LINK=EDTISAM
           FILE=TRICK
 STATUS  =  INACTIVE HOLD
 COMMAND =  FILE              DELAYED = RELEASE
```

RELEASE DELETE TFT ENTRY

The RELEASE command deletes an entry in the Task File Table (TFT; see the LINK operand in the FILE command) by means of specifying the corresponding file link name. All private volumes and private devices linked to this entry, i.e. all those requested for the associated file, are released.

Operation	Operands
{RELEASE / REL}	[link]

link — This operand specifies the file link name of the TFT entry. If "link" is omitted, the first TFT entry with the link name C'␣␣␣␣␣␣␣␣' is processed.

Notes:

- If there is more than one active file on a private volume due to be released, this volume is not released until a RELEASE command has taken effect for each active file.
- The RELEASE command is ignored if the associated TFT entry has previously been placed in the HOLD status by means of the HOLD command. RELEASE processing is not initiated until the HOLD status is canceled by a DROP command, or until the end of the job.
- If a file was exclusively reserved with the FILE operand of a SECURE command, and subsequently processed, the RELEASE command ends this reservation.

RESTART START PROGRAM AT CHECKPOINT

The RESTART command loads a program with the status it had when a CHKPT macro was encountered.

The command can be used in interactive and batch modes.

Operation	Operands
RESTART	pathname1[,page][,LOAD] [,CHECK={YES | NO}] [,ID[ENT]=chkptname] [,VSEQ={filesectno | LAST}] [,CHKPT=number] [,DUMMY={pathname2 | (pathname2,...)}] [,MONJV=jvname]

"pathname1" stands for: [:catid:][$userid.]filename1
"pathname2" stands for: [:catid:][$userid.]filename2

catid — This operand specifies the catalog identifier to which the file belongs. The file must be locally available. If this entry is omitted, the default catalog ID assigned to the user ID is assumed.

userid — This operand specifies the user identification to which the file belongs. If the operand is missing, the user ID of the LOGON command is assumed.

filename1 — This specifies the name of the file which was created by the CHKPT macro and which contains the program to be loaded.

page — This operand specifies the PAM block number at which the checkpoint records begin. These records are generated by the CHKPT macro and this number is output to SYSOUT. The system then uses this information from the checkpoint records for the restart.

LOAD — Once the program has been loaded again, it should not be started; the system reverts to command mode (cf. the LOAD command in the "Control System Command Language" manual).

CHECK=YES — This operand checks whether the internal file names of the files opened during the restart have been changed. The restart is aborted if an internal file name has changed.

=NO This specifies that an update of the internal file name is to be ignored.

ID[ENT]=chkptname

This operand specifies a 6-byte identifier for the CHKPT macro.
This identifier is output to SYSOUT together with a number (see the "page" operand) during checkpoint processing (CHKPT).

If two checkpoints with identical identifiers exist for a checkpoint file, IDENT designates the last checkpoint which has been set. The checkpoint set earlier can only be identified using the "page" operand.
If neither "page" nor IDENT has been specified, the program is started at the last checkpoint.

Note:

Specification of this operand is possible only if the checkpoint file is a disk file.

VSEQ=filesectno This designates a file section number at which the restart begins.

0 ≤ filesectno ≤ 255

This operand may only be used in the case of files which have standard labels and which have been cataloged with FSEQ=1 (cf. the FILE command).
The "page", LOAD and IDENT operands must not be used together with VSEQ.

Note:

If this operand is omitted, the default value of 0 is assumed.

=LAST This operand designates the last file section.

CHKPT=number This specifies the number of checkpoints which are passed through when restoring unusable file sections.

Possible entry: 0 ≤ number ≤ 255

This operand permits resetting at a given checkpoint and abortion after the number of written checkpoints has been reached. The checkpoint identifiers can be re-used.

Note:

If this operand is not specified, the default value of 0 is assumed.

DUMMY=filename2 The file is treated like a DUMMY file.

A maximum of 255 files can be specified.

Note:

- SYSFILE is not permissible
- The CHKPT file may be declared to be a DUMMY file only if the CHKPT operand has not been specified.

The following operand is only available to users who have the appropriate system components for job variables:

MONJV=jvname This operand specifies the name of a job variable which monitors the newly started program. The user can then use this job variable to address his program.
While processing the newly started program, the operating system sets the job variable to the value $R, $T or $A.
See also the "Job Variables" manual.

Notes:

- The job which caused the restart requests the same memory which the interrupted job had when the checkpoint was set. In addition, the checkpoint determines whether the newly started job operates in interactive mode or another mode. Files which were open when the program and operating system statuses were set are also open during restart. EAM files are, however, not restored.

  If these files are file generations, changing of the base value with respect to the CHKPT time should be avoided if at all possible. The reason for this is that the RESTART command ignores any updating of the file generation group which might have taken place between the CHKPT and RESTART times, and assumes the status valid at checkpoint time.

  It is therefore not advisable to make any further changes to the file generation groups used in the CHKPT task up until the RESTART point.

- The RESTART command may be used only if the system file SYSDTA is combined with SYSCMD and if the system files SYSIPT, SYSLST and SYSOUT have not been assigned to a file.

- The BS2000 version and the system configuration valid for the RESTART command must also be valid when the CHKPT macro is issued, otherwise the user receives an error message.

- Before the RESTART command is used, those tapes which were being processed at checkpoint time must be mounted. The checkpoint data contains the information which the system needs to reposition the tapes to the appropriate block.

- The RESTART command cannot be issued in the following two cases:

  1. If all the SYSFILE allocations (i.e. of all the opened procedures) saved in interactive mode by means of the CHKPT macro contain one or more procedures which have been interrupted, the RESTART command cannot be used in batch mode.

  2. If all the SYSFILE allocations saved in batch mode by means of the CHKPT macro contain one or more allocations of SYSOUT to a catloged file, the RESTART command cannot be used in interactive mode.

  In both of the above cases an error message is output and the job is aborted.

- In the case of insufficient class 5 memory, RESTART processing is aborted with an error message.

- It should be noted that the maximum life-span of a temporary file starts at LOGON and ends with LOGOFF.
  For this reason the user must either restore the temporary files before issuing the RESTART command or declare the temporary files as dummy files in order to perform a restart (FCB, FILE : CHKPT=DUMMY or RESTART : DUMMY=)

  If, in the course of CHKPT processing, the system determines that temporary files are being used, the left-justified byte of R15 (secondary indicator SI) is set to X'44' to warn the user, since the limited life-span of temporary files creates restrictions on the restart capability of the checkpoint which has been written.

  When the RESTART command is processed, a check is made for each temporary file of the checkpoint task to determine whether it is to be stored as a dummy file or whether a temporary file with the same name (and with the same file attributes) has been generated in the RESTART task. An error message to the effect that the temporary file does not exist may appear.

- CHECKPOINT/RESTART cannot be used to increase the life-span of a temporary file.

- After RESTART, only temporary files of the user's own task can be accessed; the user cannot access temporary files of the CHKPT task unless the latter is identical with the restart task.

- If processing of the /RESTART command is aborted with the message E305 PAM I/O-ERROR (xx), the following additional information is output:

xx	
X'04'	REQM error
X'08'	Catalog error (e.g. file does not exist)
X'0C'	A device cannot be reserved
X'10'	A file has been opened with "sharupd"
X'14'	No extension for slot segment exists
X'18'	Response "T" to message DDEE or error when writing to checkpoint file
X'1C'	VSN not the same or error on ISAM reopen
X'20'	Error in FCB
X'24'	The number of tape devices for a file is smaller than the number at checkpoint time.

Program monitoring (see also the "Job Variables" manual)

- The status indicator in the program-monitoring job variable is set to "$R" at RESTART time.
- If "jvname" is not accessible at the time the command is processed, an error message is output to SYSOUT and processing is terminated.

Network catalog management (see also the "MSCF Multiprocessor System" manual)

The RESTART command must be issued on the same system as the CHKPT macro (same processor, same home pubset, same EXEC).

Example:

The third tape of a volume series has been damaged. Reconstruction takes place by means of the RESTART command from the previous checkpoint.

```
/RESTART BAND.SICH,VSEQ=2,CHKPT=1
```

The program is terminated after the first checkpoint has been written. No update of the catalog takes place.

VERIFY RESTORE FILE

The function of the VERIFY command is to make files (also file generations and file generation groups) accessible again which were not properly closed on account of a system failure or job abortion.

The command can be used to:

- release a LOCK on a locked tape or disk file, thus making it available for general access again;
- restore a disk file. For this purpose, the catalog entry is updated, the file is closed (if necessary) and, in the case of ISAM files, the file is restored on the basis of the existing data records.

Operation	Operands
VERIFY	pathname1[,pathname2] [,REPAIR={YES | ABS | NO}] [,SUPPORT={PUBLIC | PRDISC}]

"pathname1" stands for: [:catid:][$userid.]filename1
"pathname2" stands for: [:catid:][$userid.]filename2

catid — This operand denotes a catalog identifier associated with the file. If this entry is omitted, the default catalog ID assigned to the user ID is assumed.

userid — This operand denotes the user identification. If this operand is omitted, the user's own user ID is assumed.

filename1 — This operand specifies a fully or partially qualified file name, or the name of a file generation, file generation group or temporary file.

filename2 — This operand applies only to ISAM files; it specifies the name of the file in which the ISAM file "filename1" is to be restored. If this entry is omitted, the ISAM file is restored in a work file provided by the system. In the case of an ISAM file on private volumes, where the data and index blocks are located on different disks, the file will be restored only if "filename2" has been specified. "filename2" must be different from "filename1" and be a fully or partially qualified file name, i.e. the entry is ignored if it is the name of a file generation or of a file generation group.

Specification of a temporary ISAM file is permissible.

Note:

If a user ID other than the user's own is specified in a path name, the file must be shareable.

REPAIR — The following operands are used when restoring files:

=YES

PAM: The file is unlocked. If the file is marked as an open file, a privileged close operation is performed. The pointer to the last PAM block remains unchanged.

SAM: The file is unlocked. If the file is marked as still being open, the pointer to the last PAM block is set to the highest page containing data. The file is closed.

ISAM: The file is unlocked and, if it is marked as being opened, it is restored.

=ABS

PAM: The file is unlocked, after which the pointer to the last PAM block is set to the highest PAM page containing data.

SAM: The file is unlocked. Even if the file is not marked as being open, the pointer is set to the highest PAM page number containing data. The file is closed.

ISAM: The file is unlocked and restored.

=NO

PAM: The file is merely unlocked (removal of the entry from the table of locked files). Thereafter the file is still not regarded as being closed, i.e. it will be listed in response to FSTATUS, STATE=NOCLOS, and with VERIFY ..., REPAIR=YES, it will be regarded as a file to be repaired.

SAM: The file is merely unlocked.

ISAM: The file is merely unlocked and, if it is still marked as being opened, a privileged close operation is performed and the pointer to the last PAM block is set to the highest PAM page number containing data.

SUPPORT=PUBLIC — This operand selects files on public volumes for restoration.

=PRDISC — This operand limits restoration to files located on private volumes. The SUPPORT operand is meaningful only when "filename1" specifies a partially qualified file name or the name of a file generation group.

Notes:

- Following the message "FILE NOT ACCESSIBLE DUE TO SPACE PROBLEM" the file can no longer be made available with VERIFY. In this case it is possible to give only the command /ERASE␣...,CATALOG (even for PUBLIC files).

- In order to be able to cancel a file lock, the job which caused the lock must have been terminated by the system with the console message "TASK PENDED INDEFINITELY".

  For files declared as exclusively reserved via the SECURE command, the user can cancel the lock; in the case of disk files, only the system administrator is authorized to do so.

- If file access was interrupted while data buffers were located in main memory, the last modifications made may be missing from the restored file. This is because a buffer is not written to external storage until it is full.

Notes on the restoration of ISAM files:

- If the entry "filename2" is omitted for an ISAM file on public volumes, the file is restored in a work file provided by the system. "filename1" is then deleted, without the DESTROY entry (see the ERASE command), and the work file is renamed "filename1".

- If the entry "filename2" is omitted for an ISAM file on private volumes, the file is restored in a temporary work file on public volumes. The work file is subsequently copied to the file "filename1" and the work file deleted, with the DESTROY entry (see the ERASE command). This procedure can be extremely time-consuming, and thus it is more convenient to specify "filename2".

- If a file "filename2" is defined in the VERIFY command, the ISAM file "filename1" is restored on this file, "filename1" remaining unchanged. If "filename2" is located on private volumes, or if "filename1" is a file on private volumes, the user must catalog the file "filename2" before issuing the VERIFY command. If the data and index blocks are located on separate volumes, the user must also allocate space for "filename2" via a FILE command.

- With ISAM files, the data buffers are written back to the disks as soon as a new data block has to be fetched into main memory. This may result in the last modifications made being omitted from the restored ISAM file. If, however, WROUT=YES was specified in the FILE command or macro, the effect of the error is negligible, since the data buffer is written back to the disk after each modification.

- Records having the same index and containing the same data are included in the restored file once only.

- No space is reserved in the data blocks of the restored file for subsequent additions; this corresponds to PAD=0 in the FILE command.

- ISAM files whose data and index blocks are on separate private volumes can be restored with the VERIFY command only when BLKSIZE=STD is specified.

- If an ISAM data block contains data which cannot be assigned to any defined record, the whole block is saved to a PAM file bearing the name "S.filename1.REPAIR". Once VERIFY processing has been completed, this file is available to the user for his own attempts at restoration. If the file name gets to be too long, filename1 is truncated accordingly.

- As a copy of the file is created when restoring ISAM files and this copy is part of public space, the user must make first make sure that there is sufficient public space available. If "filename1" is a temporary file, it is advisable to specify "filename2" as a temporary file too.

3.2 DEVICE MANAGEMENT

SECURE[-RESOURCE-ALLOCATION] REQUEST RESOURCES

The SECURE[-RESOURCE-ALLOCATION] command reserves resources required by a job during its execution. This reservation ensures that a subsequent request for access to these resources will not be rejected by the system.

These resources may include:

- disks
- files, file generations and file generation groups.

Reservation of the file also results in reservation of the disk, and in turn, reservation of the devices.

An existing reservation can be deleted by means of:

- The RELEASE command:
  This cancels the reservation of the file and, if necessary, of the associated disk. The same device reservation as before is maintained. If disks or devices assigned to the file are also implicitly reserved by means of other files or disks, these are not released until all system references have been deallocated.
- Renewal of the SECURE[-RESOURCE-ALLOCATION] command
- The WHEN command:
  This command has the same function as SECURE[-RESOURCE-ALLOCATION] without the need for specifying the operand.
- End of job (LOGOFF).

The SECURE[-RESOURCE-ALLOCATION] command can be used in both interactive and batch mode.

The following two commands are available for reserving resources:

SECURE

SECURE-RESOURCE-ALLOCATION

However, the functional scope of the SECURE command, whose syntax is compatible with Version 7.5, is restricted. As far as internal processing is concerned, there is no difference between the two commands, i.e. the notes on the SECURE-RESOURCE-ALLOCATION command also apply for SECURE.

SECURE format

Operation	Operands
SECURE SEC	[VOL=(vsn/type,...)] [,FILE=(pathname[/EX],...)]

VOL=(vsn/type,...)

This operand specifies the volume serial number (6 bytes long) and the device type for the volume to be secured. Volumes reserved are shareable.

The entries in the VOL operand must be enclosed in parentheses, and individual entries separated by commas. Up to 16 entries are possible.

The following disk storage units may be specified:

D3455 or D5881
D3465 or D5882
D3468
D3470 or D5804
D3475
D3480

FILE=

This operand reserves the specified file, file generation or file generation group.

If more than one file name is specified in the FILE operand, they must be separated by commas and enclosed in parentheses. Up to 48 entries are possible.

=pathname

"pathname" stands for [:catid:][$userid.]filename

=catid

This operand specifies the identifier of the catalog to which the file belongs.
If this entry is omitted, the default catalog ID assigned to the user ID is assumed.

=userid

This operand specifies the user ID to which the file belongs. If no entry is made, the user's own ID is assumed.

=filename

This entry specifies the fully qualified name of a file or file generation which has already been cataloged.

=(filename[EX],...)

This operand specifies exclusive reservation of a file, file generation or file generation group; i.e. in addition to any existing implicit reservation of volumes and devices, a file lock is effected, thus preventing any other job from accessing the file/files.

If "filename" is the name of a file generation, the appropriate reservation is performed, and the associated file generation group is locked.

If "filename" refers to a file generation group, the entire group is blocked against foreign access.

If one of the requested resources is not available, a distinction is made between interactive mode and batch mode in subsequent processing:

Interactive mode:	The command is rejected with an appropriate message
Batch mode:	The job is put into a wait queue, the SECURE-QUEUE; there is no restriction on the waiting time, which lasts for as long as it takes to make all requested resources available simultaneously.

SECURE-RESOURCE-ALLOCATION format

Operation	Operands
SEC-RES	[DISK={*NO | (VOLUME=vsn,TYPE=device [,ALLOCATION={EXCLUSIVE | SHARED}])L}]
	[,FILE={*NO | (NAME=pathname[,ALLOCATION={EXCLUSIVE | SHARED}])L}]
	[,WAIT=([TIME={TASK-STD | n}])

The flag "L" indicates that the operand can be specified in list form, e.g. (op1,op2,...).

DISK operand

VOLUME=vsn This operand specifies the vsn of the device which is to be reserved (maximum of 6 characters).

TYPE=device This operand specifies the device on which the disk is to be operated. "device" may be any of the following:

D3455
D3465
D3468
D3470
D3475
D3480

ALLOCATION= This specifies the mode of reservation used to reserve the disk.

=EXCLUSIVE
Exclusive reservation means that the resource cannot be used by other jobs.

=SHARED
This is the default value, specifying that other jobs may also use the device.

FILE operand

NAME=pathname "pathname" stands for [:catid:][$userid.]filename

catid This specifies the identifier of the catalog to which the file belongs. If this entry is omitted, the default catalog ID assigned to the user ID is assumed.

userid This specifies the user ID to which the file belongs. If the entry is omitted, the user's own ID is assumed.

filename This operand specifies the fully qualified name of a file or the name of a file generation which has already been cataloged. Both system and user files can be reserved; the only thing that is verified is whether or not the file or file generation exists. If it does not exist, the job is not placed in the device queue; an error message is issued instead. In batch mode, a branch is made to the next STEP, ENDP, ABORT, ABEND or LOGOFF command (cf. the "Control System Command Language" manual).

All disks belonging to the file are reserved and mounted. Disks are always regarded as being shareable.

ALLOCATION= This operand specifies the reservation mode of the file.

=SHARED
Both the file and the corresponding disks or devices are reserved and shareable. The file can be opened by other jobs (in both read and write mode), i.e. only implicitly required disks and devices are reserved.

=EXCLUSIVE
The file can only be opened by the job which reserved it. Corresponding disks and devices are still shareable as far as reservation is concerned.
Exlusive reservation of the file is possible only if it is not being used.
If the file is a file generation group, the entire file generation is locked against access by other users.

WAIT operand

TIME= This specifies the maximum period of time the job will wait for execution of the reservation request.

=TASK-STD A distinction is made between interactive mode and batch mode.

Interactive mode: The command is rejected if the request for reservation cannot be met.

Batch mode: In principle, the system waits for mounting and release of resources for the length of time specified. If the requests have not been fulfilled and the waiting period has expired, a branch is made to the next STEP, ENDP or LOGOFF command.

=n This operand specifies accuracy, in minutes. The maximum value is 43,200 seconds (= 12 hours). If less than 180 seconds are specified, the command is rejected if mounting of a disk is required or if confirmation on the part of the operator is required.

Notes:

- The SECURE command is allowed only when no program is loaded.
- Other users' files can be reserved only if they were entered in the catalog with SHARE=YES (see the CATALOG command).
- Files under the ID of the system administrator can only be exclusively reserved by jobs running under this ID.
- Each SECURE[-RESOURCE-ALLOCATION] command (including those without operands) first releases all previously reserved private resources (implicit RELEASE).
- If a volume which has not yet been mounted is reserved using the SECURE command, a MOUNT message appears on the console.
- A request for a volume made via the FILE operand in the SECURE command is referred to as **implicitly** requested, otherwise it is referred to as **explicitly** requested.
- Any attempt to reserve a public volume explicitly will be rejected.
- A resource is considered exclusively reserved if no other job can use it while it is reserved. If the system guarantees free access while it is reserved without prohibiting other jobs from using it, the resource is termed shareable.
- If one of the resources requested cannot be secured, no reservation is carried out at all. Depending on the type of job and the specified WAIT operand, if any (SECURE-RESOURCE-ALLOCATION command only), either an appropriate message is issued by the system or the job is held in the SECURE queue until all the resources requested are available for the job or until the specified waiting time has expired. However, a branch is made to the next STEP or LOGOFF command if the device or number of devices requested are not available.
- If a SECURE command is rejected (resources not available, for example), all the volumes previously allocated to the job are released.
- If the operator refuses the request for a resource (MOUNT message on the console), no reservation of any sort is undertaken and, in batch mode, control is passed to the next STEP or LOGOFF command (see the "Control System Command Language" manual).
- If a file which extends over several private volumes is reserved, all the volumes are reserved.
- The system rejects a SECURE[-RESOURCE-ALLOCATION] command if the calling job has opened files on exclusively reserved private volumes.

- Files on private volumes can be opened in the following cases:

  - if the SECURE command originates from a procedure file which is itself located on private volumes;

  - if a file on private volumes has previously been assigned with the SYSFILE command.

  Such a file should be treated as a resource which, when the SECURE [-RESOURCE-ALLOCATION] command is processed, must first be released before new resources can be reserved.

- The reservability of a disk is dependent on the values given in the USER-ALLOCATION, SYSTEM-ALLOCATION and OPERATION-CONTROL operands and in the SET-DISK-DEFAULTS or SET-DISK-PARAMETER operator command. These values can be interrogated by the user by means of either the /SHOW-DISK-STATUS command or an appropriate DSTATUS macro.

- Shareable private disk (SPD):

  A SECURE FILE command for files which reside on an SPD is entered in the F1 label of the private disk. Other processors accessing this SPD respect this SECURE command.

- If a pubset specified by means of "catid" in the FILE operand is not available locally and not contained in the RFA table of the job, the command is rejected with an error message. It is therefore not possible to wait for the local availability of a pubset.

<u>Remote file access (see also the "RFA" manual)</u>

- The SECURE command cannot simultaneously reserve local and remote volumes.

- The local TFT contains entries for all remote files that are processed. However, this does not include remote files that are exclusively reserved, or remote resources after the /RELEASE command has been issued.

- Files can also be reserved in a remote system via RFA; however, this command may only include files and all of these must be part of the same system.

- To prevent a deadlock situation from occuring, a SECURE command without operands is passed on to all RFA partner tasks (unless the task is itself an RFA partner task) during the release phase at the start of command processing.

- If an RFA partner task is running on a system less than or equal to Version 7.6, the SECURE command must be issued.

Example:

A batch job reserves the file X.VORHER using the command:

/SECURE FILE=X.VORHER/EX

Thereafter any interactive job attempting to read this file will receive the error message "LOCK ERROR OD99".

SHOW-DEVICE-STATUS REQUEST RESERVATION AND MONITORING INFORMATION

This command provides information on the reservation and monitoring of volumes. The information output pertains to volumes which are physically online (in contrast to that output by the SHOW-DISK-STATUS command).
If no volume is online for a device, the display indicates which volume is to be mounted on the device.

The user obtains an output record only for the devices for which his job has made a reservation.

There is no restriction in respect to information output using the INFORMATION=SUMMARY operand.

Operation	Operands
{SHOW-DEVICE-STATUS SH-DEV}	{UNIT={mn (mn1,mn2,...)} TYPE=device-type[,ATTR[IBUTE]={ALL attribute}] FAM[ILY]=device-family[,ATTR[IBUTE]={ALL attribute}] ATTR[IBUTE]={ALL attribute}} [,INF[ORMATION]={STD SUM[MARY] ALL SHORT}]

UNIT=mn
: This operand specifies one or more hardware units on which information is output; the units are specified via their mnemonic device names (2 alphanumeric characters). A maximum of 26 units may be specified.

TYPE=device-type
: Information is output on all devices of the specified device type. The following device types are possible:

D3455, D3465, D3468, D3470, D3475, D3480 D348E	Disk Storage Units

FAMILY=device-family
: Information is output on all devices which belong to the specified family of devices. Possible device families are:

DISK	Disk devices

ATTRIBUTE=ALL Information is output on devices with all possible attributes.

=attribute
Information is output on all devices with the specified attribute. Possible attributes are:

ATT[ACHED]
DET[ACHED]
DET[ACH]-P[ENDING]
FREE
DMS
EXCL[USIVE]
PUB[LIC]

INFORMATION= Specifies the type of information desired.

=STD A standard output record is output (see output formats). When TYPE or FAMILY is specified, only default information is supplied.

=SUMMARY
A summary of configuration and reservation statuses is generated.

=ALL Output of STD and SUMMARY.

=SHORT
Provides a brief description of STD without header and without SUMMARY records of the specified family or of the specified type.

Output format

A header and, for each device, a continuation line with the values are output. Depending on the specification in the INFORMATION operand, four output formats are possible. See Table 3-1 for the meaning of the output columns and possible values.

1. Output for INFORMATION=STD

Header:

MNEM	DEV-TYPE	CONF-STATE	POOL	VSN	DEV-A	PHASE	ACTION

Values:

mn	device type	short configuration state	pool attribute	vsn	device allocation state	volume phase	action state

Note:

During a REMOUNT procedure, a vsn can simultaneously appear in two output records, namely in the record of the device on which the volume is physically online and in the record of the device on which the volume is to be mounted.

2. Output for INFORMATION=SUMMARY

   Header:

   DEV-TYPE AVAIL PRE-/IN-USE RES-BY-MN RES-BY-TYPE ATT DET DET-P

   Value line:

dev type	# of still available devices	(*)	# of devices reserved with mnemonics	without	# of devices that are attached,detached detach-pending

(*) : Number of devices with the status PREMOUNT,MOUNT(ing), IN-USE

3. Output for INFORMATION=ALL

   Formats 1 and 2 are output.

4. Output in brief for INFORMATION=SHORT

   Value lines without header and with the following elements:

   mn , letter of conf state , short device alloc state , spd indicator

   Values:

   letter of conf state = A(ttached) , D(etached)
   P (detach-pending)

   spd indicator = *, provided that SPD features are present

   short device alloc state = tsn of the exclusively reserving task
   FREE , DMS , PUB(lic)

   This is followed by output of INF=SUMMARY.

SHOW-DISK-DEFAULTS INTERROGATE DEFAULT VALUES FOR DISK PARAMETERS

The SHOW-DISK-DEFAULTS command provides information concerning the default values for the DISK parameters which have been set using the operator command SET-DISK-DEFAULTS.

Operation	Operands
{SHOW-DISK-DEFAULTS SH-DISK-DEF}	

Output format

One header and one value line are output. See Table 3-1 for the meaning of the output columns.

Header:

```
ASSIGN-TIME     USER-ALLOCATION    OPERATOR-CONTROL
```

Value line:

```
USER ,          NO , ALL           NO , ALL
MOUNT ,         EXCL(usive)  ,     EXCL(usive)  ,
OPERATOR        SHARE              SHARE
```

SHOW-DISK-STATUS INTERROGATE RESERVATION AND DISK PARAMETERS

The SHOW-DISK-STATUS command provides information on reservation and DISK parameters as well as on volume monitoring for the specified disks. The information which is output refers to the reserving volume, irrespective of which volume is physically online (as opposed to the SHOW/DEVICE-STATUS command).

The user only obtains an output record for those disks for which his job has made a reservation.

Operation	Operands
{SHOW-DISK-STATUS SH-DISK}	{{UNIT={mn (mn$_1$,mn$_2$,...)}}} [{{VOL[UME]={vsn (vsn$_1$,vsn$_2$,...)}}}] {ATTR[IBUTE]={ALL attribute}} [,INF[ORMATION]={STD PAR[AMETER]}]

UNIT=mn — This operand specifies the disk devices on which information is output by means of their mnemonic device names (2 alphanumeric characters). Up to 26 units can be specified.

VOLUME=vsn — This operand specifies the disks on which information is output by means of their volume serial numbers (maximum of 6 characters). Up to 8 vsn's may be specified.

ATTRIBUTE=ALL — Information is output on disks with all possible attributes (default value).

=attribute
Information is output on all those disks that include the specified attribute, which may be one of the following:

FREE
EXCL[USIVE]
SHARE
} volume allocation status

PUB[LIC] — device allocation status

ONLINE
MOUNT
IN-USE
} volume phase

CAN[CELLED] NO-DEV[ICE] REC[OVER] DISMOUNT UNLOCK SVL-UPD[ATE]	action state
DMS SPECIAL	use mode
STD N[ON]-STD BS1000	label type

INFORMATION= Specifies the type of information desired. This operand is valid only in connection with the UNIT OR VOLUME specification. Only INFORMATION=STD is possible for ATTRIBUTE.

=STD For each disk, a record with global reservation and monitoring information is output.

=PARAMETER
The parameters set by the operator are output. Parameter values which have been set as the default values are identified by "(D)". This operand cannot be used to interrogate information on public disks.

Output format:

A header and, for each specified disk, a value line are output. Depending on the specification in the INFORMATION operand, five output formats are possible. See Table 3-1 for the meaning of the output columns and possible values.

1. Standard output record

   Header:

MNEM	VSN	USE	LABEL	DEV-A	VOL-A	PHASE	ACTION

   Values:

mn	vsn	use mode	label type	device allocation state	volume allocation state	volume phase	action state

2. Output record for INFORMATION=PARAMETER

   Header:

MNEM	VSN	TYPE	ASS-TIME	USER-ALLOC	OP-CTL	SYS-ALLOC	ACCESS

   Values:

mn	vsn	device type	assign time	user allocation type	operator access control	system allocation mode	access mode

Notes:

1. In the case of public disks, only mn and vsn are output when the parameter INF=PAR is specified.

2. If more than one disk with the same vsn exists within a system, only one record is output when INF=PAR is specified.

SHOW-MOUNT-PARAMETER INTERROGATE MOUNT SPECIFICATIONS

SHOW-MOUNT-PARAMETER provides information on the specifications made by the operator using the MODIFY-MOUNT-PARAMETER command relating to the mounting and demounting of volumes.

Operation	Operands
SHOW-MOUNT -PARAMETER SH-MOUNT-PAR	

Output format

One header and one value line are output. See Table 3-1 for the meaning of the output columns.

Header:

TAPE-MOUNT DISK-MOUNT ALLOCATE-TAPE UNLOAD-RELEASE-TAPE

Value line:

YES , NO YES , NO YES , NO YES , NO

SHOW-RESOURCE-ALLOCATION INTERROGATE TASK RESERVATIONS AND OPEN OPERATOR ACTIONS

The SHOW-RESOURCE-ALLOCATION command provides information on reservations and on operator actions which are still open for the user's own job.

Operation	Operands
[SHOW-RESOURCE -ALLOCATION SH-RES]	[[TSN=[*OWN[-TSN] tsn] MON[JV]=monjv ITN=X'itn']] [,ID[ENTIFICATION]=[JOB[-NAME] USER[-IDENTIFICATION]]] [,INF[ORMATION]=[RES[OURCES] ACT[IONS]]]

TSN=*OWN-TSN This operand provides information on the user's own tsn (default value).

=tsn Designates a job by means of the task sequence number. The user may specify only his own tsn.

JOB-NAME=job-name
This operand designates a job by means of its job name.

ITN=X'itn' This designates a task by means of its internal task number. When X'0' is specified, the user obtains information on his own ITN.

IDENTIFICATION= This operand controls the occupancy of output field NAME/ID.

=JOB-NAME
The job name is entered in the field NAME/ID (default value).

=USER-IDENTIFICATION
The user ID is entered in the field NAME/ID.

INFORMATION= This specifies the type of information desired.

=RESOURCES
A record concerning every resource reservation for the specified task is output (default value).

=ACTIONS
A record is output for each open operator action for a volume of the specified task.
Operator actions include:
Mounting, mounting a write-enable ring, rectifying INOP, advance mounting, remounting, etc.

Output format

One header and one value line are output. Two output formats are possible, depending on the specification in the INFORMATION operand. See Table 3-1 for the meaning of the output columns and possible values.

1. Output for INFORMATION=RESOURCES

   Header:

   MNEM DEV-TYPE VSN VOL-A TSN NAME | ID PHASE

   Values:

   mn device allocation type vsn volume allocation state tsn cf. IDENTIFICATION operand volume phase

2. Output for INFORMATION=ACTIONS

   Header:

   MNEM DEV-TYPE VSN VOL-A TSN NAME | ID PHASE

   Values:

   mn device allocation type vsn volume allocation state tsn cf. IDENTIFICATION operand volume phase

Meaning of the output columns in conjunction with the SHOW commands

Keyword	Used in command	Meaning
ACCESS	SH-DISK (INF=PAR)	= READ: Defines how the disk is used in PPD mode (PPD = Protected Private Disk; chargeable product). In this mode of operation the data on the disk is read only, no write access is allowed. = WRITE: Both read and write access are permitted for the disk. = ALL: The final ACCESS value is not defined until the disk is reserved; this is dependent upon the generation specification of the device on which the disk is mounted: *POOL=NO|SH results in the setting of ACCESS=WRITE *POOL=SW results in the setting of ACCESS=WRITE This value is independent of the position of the WRITE INHIBIT switch.
ACTION	SH-RES SH-DEV SH-DISK	Indicates which reservation or mount procedure started by the volume monitoring facility is presently being executed. This may be initiated by the following actions: - Operator intervention (inadvertent demounting of a volume still being used) - Commands (DETACH, MOVE ...) - Device Error Recovery (DER), e.g. in the event of INOP - User request (MOUNT message for volumes not yet mounted).

Meaning of the output columns

Keyword	Used in command	Meaning
ACTION continued		The following status conditions may occur: CANCELLED: A tape or disk is permanently locked against use; the cancelation can no longer be changed. DISMOUNT: Either a REMOUNT message is pending for the same volume on another device, or a REMOUNT or MOUNT message is pending for another volume on the same device. INOP: The device is temporarily unavailable (inoperable). MOUNT: The response to a MOUNT message is pending for the volume in question. NO ACTION: No interrupt. NO DEVICE: Due to a previous reconfiguration command (DET, REM) there is no longer any device allocation for a volume which has been reserved. POSITION: A tape being used is repositioned. PREMOUNT: The response to a PREMOUNT message is pending for the volume in question. RECOVER: Interrupt processing is taking place for the volume in use; the exact type of processing is not specified. REMOUNT: A remount operation is being performed. SVL-UPDATE: The system reservation log is being stored on disk. UNLOCK: An UNLOCK job for a system ID stored in the SVL is being performed. WP-MISSING: Either the write-enable ring is to be mounted for a tape or the write protection for a disk is to be cancelled.

Keyword	Used in command	Meaning
ASS[IGN]-TIME	SH-DISK (INF=PAR) SH-DISK-DEF	In the case of private disks, this defines the time at which a disk is reserved or released in USE=DMS mode (SH-DISK INF=PAR). With SH-DISK-DEF the value for ASSIGN-TIME defines the time of reservation or release for all the disks for which this value was not specifically set.
ATT	SH-DEV (INF=SUM)	Shows the number of devices in the ATTACHED state which belong to the device type specified in the output (independent of the reservation).
AVAIL	SH-DEV (INF=SUM)	Shows the number of devices still free (available) for the device type defined in the output.
CONF-STATE	SH-DEV	Indicates the configuration status (ATTACHED, DETACHED, DET-PENDING) of the specified device, from which the availability of said can be deduced.
DET	SH-DEV (INF=SUM)	Indicates the number of generated devices of the specified type which are not available as a result of their configuration status (DETACHED).
DET-P	SH-DEV (INF=SUM)	Number of devices of the type in question which the system still needs to fulfil user requests and which are to be detached after they have been released. New reservation of these devices is no longer possible.

Meaning of the output columns

Keyword	Used in command	Meaning
DEV-A	SH-DEV SH-DISK	Provides information on the type of device reservation: FREE : The device is not yet reserved, i.e. it is still free. DMS : The device in question is implicitly reserved by the DMS application on the mounted private disk. PUBLIC: The device is implicitly reserved by an active public disk which has been mounted. tsn : TSN of the job exclusively reserving the device. Either: - the TSN is requested by means of SECURE UNIT, or - in the case of disk devices, the job making the reservation uses the assigned disk for a USE-SPECIAL application (PHASE=IN-USE or MOUNT).
DEV-TYPE	SH-RES SH-DEV	TSOS type (D3465, UM1662,...) of a device or device definition which the user has specified for his device reservation (T800, T1600,...)
LABEL	SH-DISK	Defines the type of volume label for a volume (cf. SVL). The following values are possible: STD: The volume has standard labels; BS1000: The disk has BS1000 labels; TAPE-MARK: The tape begins with a tape mark; NON-STD: The label has none of the three attributes listed above.
MNEM	SH-RES SH-DEV SH-DISK	This field defines the mnemonic name of the device as specified during generation.
NAME/ID	SH-RES	Provides information on the job name of the job addressed or on the user ID under which the job runs.
OP-CTL OPERATOR-CONTROL	SH-DISK (INF=PAR)	Specifies whether the operator wishes to be informed of the first time disks are reserved by the tasks (with the option of rejecting these reservation requests).

Keyword	Used in command	Meaning
PHASE	SH-DEV SH-DISK SH-RES	ONLINE: The volume is mounted but not reserved; PREMOUNT: The volume is reserved; a device reservation exists for future use; MOUNT: The volume has already been reserved but still has to be secured by the operator; IN-USE: The volume is released for use (except when ACTION=CANCELLED is specified). Volume monitoring takes place for the statuses PREMOUNT and IN-USE (a volume is always monitored if a valid reservation exists for it). In such a case the volume monitoring facility is responsible for the following functions: - it ensures allocation of a device for tapes in the PREMOUNT status; - it requests the operator to make a volume available again when the volume is considered to be reserved but is currently not accessible (INOP); - it sees to it that a NO-DEVICE status is cancelled as soon as a device of the required device type is free; - it initiates automatic repositioning of tapes in the event of an operator error (e.g. if he dismounted the wrong tape device).

Meaning of the output columns

Keyword	Used in command	Meaning
POOL	SH-DEV	Defines the availability of a device in respect to several systems. Three definitions are possible: 1. POOL=NO: This device is only available from within the user's own system. Access to a volume mounted therein is not possible when attempted from another system. 2. POOL=SH: As a rule, this device is generated for several systems. Its hardware supports parallel use of more than one system (device with multiprocessor connection). A disk which is mounted on a device with this generation feature is generally operated as a shareable private disk (**-generation with UGEN). 3. POOL=SW: As a rule, this device is generated for several systems (x-generation with UGEN). In the case of disk devices: the device likewise has a multiprocessor connection; however, in contrast to POOL=SH, a private disk mounted on this device is generally system-exclusive (no SPD operation). If a PPD is used, the private disks mounted on these devices are generally only available for read access.
PRE-/IN-USE	SH-DEV (INF=SUM)	Specifies the number of devices of the defined type which are implicitly reserved by means of volumes of the corresponding phase.
RES-BY-MN	SH-DEV (INF=SUM)	Specifies the number of devices of the type in question which were reserved by a user by means of SEC-RES UNIT=mn.
RES-BY-TYPE	SH-DEV (INF=SUM)	Specifies how many free devices of the specified type are required to handle previously acknowledged reservation requests (SEC-RES TYPE=3465,NUMBER=2)

Keyword	Used in command	Meaning
SYS-ALLOC	SH-DISK (INF=PAR)	Specifies the mode of operation for use of a private disk by the user's own system with regard to other systems when USE=DMS is specified. Possible operating modes: EXCLUSIVE: Other systems are prohibited from using the disk. SHAREABLE: Other systems may also access the disk (SPD operation). Synchronization with the other systems is performed with respect to the use of files and storage space; Catalog locks are kept on disk.
TSN	SH-RES	Specifies the 4-digit task sequence number for generating a job.
TYPE	SH-DISK (INF=PAR)	Defines the device type of the volume from which information is requested. Apart from a request on the part of the user (SECURE,FILE,...), the device type specification may also depend on the following: 1. Online event: As a result of an activation interruption the volume is assigned to a device whose DEVICE-TYPE specification then determines the device type of the volume in the event of a vsn request. 2. SET-DISK command: As a result of predefinition of the device type by means of this command, the type can be defined even before a volume is reserved.

Meaning of the output columns

Keyword	Used in command	Meaning
USE	SH-DISK	Provides information: - on the operating mode of a mounted volume; - on the resulting degree of monitoring; - on the extent of checks made when allocation is performed by the monitoring facilities. The following values are possible: DMS: The volume is reserved by one or more DMS applications. Only readable volumes are accepted for allocation, i.e.: - Disks with or without standard labels can be processed, but they must be uniquely identifiable. The default task reservation mode for USE=DMS is task-shareable for private disks and task-exclusive for tapes. Each operator intervention during PHASE=IN-USE leads to REMOUNT/RECOVER, and to repositioning in the case of tapes. It is ensured that only one volume with the same vsn is reserved for DMS usage mode. SPECIAL: The volume is reserved by a special application (privileged application, e.g. VOLIN, INIT, testing and diagnostic programs, FDDRL, etc.) The task and system reservation mode is EXCLUSIVE. Checks made during allocation (VOLIN,INIT) or monitoring functions such as repositioning or MOVE (online FDDRL controls this by itself) can be switched off by means of the special application. No verification of vsn uniqueness takes place. WORK: The mounted tape is used as a work tape (it is made available to the DMS user for processing work files).

Keyword	Used in command	Meaning
USER-ALLOCATION	SH-DISK (INF=PAR) SH-DISK-DEF	Specifies which reservation requests made by the user (task-shareable, task-exclusive) are permissible for a private disk for which USE=DMS has been specified.
VOL-A	SH-RES SH-DISK	In the case of current public disks, this field merely indicates whether the mounted disk is the SYSRES disk, a PAGING disk or a conventional disk on public without special attributes (PUBLIC). In the case of private volumes, it provides information on the volume's reservation as made by a user: PAGING: The disk is part of the reserved pubset and is used for paging. PUBLIC: The disk is part of the reserved pubset of the reserved system. SYSRES: Public disk of the home pubset, on which catalog TSOSCAT begins. FREE: At the moment no user is accessing the volume. EXCL: The private volume is exclusively assigned to a user job (for which period of time other users cannot work with it). SHARE: The private disk is already reserved by one or more jobs. Requests from further users are permitted.
VSN	SH-RES SH-DEV SH-DISK	"Name" of a volume: i.e. the volume serial number defined when a volume is initialized (VOLIN, INIT). If the volume does not have a readable label or if the request for the volume did not include a vsn, synonyms may also be output. The following values are possible: <vsn>: The vsn of a volume defined with VOLIN or INIT; see the VOLUME operand of the FILE command in the "Control System Command Language" manual; *UNKNO: The volume has no BS2000 standard label; *SCRAT: The request for the volume did not include a vsn (e.g. in the case of tapes, a FILE command without a VOLUME operand).

Table 3-1: Meaning of the output columns used for the SHOW commands

4 MACROS WITH COMMAND FUNCTIONS

This chapter contains the formats of those macros which have command functions.

Conventions relating to BS2000 macros can be found in Appendix A.1.

CATAL PROCESS CATALOG ENTRY ‹TYPE S›

The CATAL macro creates or updates the catalog entry for a file, file generation group or temporary file.

Format 1 for files

Operation	Operands
CATAL	pathname1[,filename2] [,STATE={N[EW] | U[PDATE]}][,ACCESS={WRITE | READ}][,SHARE={NO | YES}] [,WRPASS={NONE | password1}][,RDPASS={NONE | password2}] [,EXPASS={NONE | password3}][,RETPD=days][,BACKUP={A | B | C | D | E}] [,LARGE={YES | NO}][,DESTROY={YES | NO}][,AUDIT={SUCC | FAIL | ALL | NONE}]

Format 2 for file generation groups

Operation	Operands
CATAL	pathname1[,filename2] [,STATE={N[EW] | U[PDATE]}][,ACCESS={WRITE | READ}][,SHARE={NO | YES}] [,RDPASS={NONE | password1}][,WRPASS={NONE | password2}] [,RETPD=days][,GEN=max] [,DISP={CYCLE | REUSE | DELETE | KEEP}][,BASE={absbas | relbas}][,FIRST=n] [,DEVICE=device][,VOLUME=vsn][,BACKUP={A | B | C | D | E}] [,LARGE={YES | NO}][,DESTROY={YES | NO}][,AUDIT={SUCC | FAIL | ALL | NONE}]

For a description of the operands see the CATALOG command.

Programming note:

Register 15 is set to 0 upon successful completion of this macro. The error codes for unsuccessful execution are specified in the IDEMS macro (Appendix A.3).

CHNGE MODIFY TFT ENTRY (TYPE S)

The CHNGE macro changes the file link name of an entry in the Task File Table. All other TFT entries remain unaffected.

Operation	Operands
CHNGE	[link1],link2

For a description of the operands see the CHANGE command.

Programming notes:

- The CHNGE macro is meaningful only for a TFT entry which was previously created by means of the FILE command/macro (not with OPEN).

  The reason for this is the different handling of the TFT entry when the file is reopened (see section 5.1, "Opening Files").

- The following return codes are set in register 15:

  00 - request successfully completed.

  The error codes for unsuccessful execution are specified in the IDEMS macro (Appendix A.3).

COPY COPY FILE (TYPE S)

The COPY macro copies files, file generations, file generation groups or temporary files without modifying them.

Operation	Operands
COPY	pathname1,pathname2[,SAME][WRITE={REPLACE | NEW}]

For a description of the operands see the COPY command.

Programming note:

The following return codes are set in general register 15:

00 - request successfully completed.

The error codes for unsuccessful execution are specified in the IDEMS macro (Appendix A.3).

To avoid a collision with the Assembler statement "COPY", the following command must be issued prior to the COPY macro:

COPY OPSYN

DSTATUS OUTPUT DATA ON PERIPHERAL CONFIGURATION (TYPE S)

The DSTATUS macro permits access to data provided by the information service facility (NKD) of the Nucleus Device Management (NDM). The data provides information on:

- the resources occupied by a task;
- the reservation and availability status of resources for devices, device types, disks or tapes;
- the structure of the configuration;
- the device queue.

The DSTATUS macro offers two separate basic functions:

Function 1: Writing of output records.

Output records contain information on the reservation and availability status of the specified devices, hardware units, resources, or of the overall configuration. The records are entered in an area of class 6 memory. This output area is comprised of an output control record and the relevant output records. The start address of the area is transferred in register R1.
The user issuing the macro is responsible for returning the storage area (RELM). The length specifications are to be taken from the output control area. When returning the storage area, care must be taken to ensure that:

- the start address of the output buffer is aligned on the page boundary
- the length is converted from bytes to pages.

Function 2: Generation of a DSECT for output records.
With the aid of the symbolic name of the DSECT, the user can access the contents of output records or obtain information on the structure of output records.

The following restrictions apply to non-privileged users:

- Task records are only output for tasks under the user's own ID
- The user does not obtain any information on device queues.
- Information pertaining to devices, disks and tapes is output only if the calling task has the appropriate resources at its disposal.

Macro formats and operand descriptions

Function 1: Writing of output records

Macro	Operands
DSTATUS	[TASK={OWN | addr | (r)}]
	[,DEVICE={ALL | ({FC | TT | MN}, {addr | (r)})}]
	[,DISC={MONITORED | SCHEDULED | ({VSN | MN}, {addr | (r)}, {L,[TASKS] | S})}]
	[,CONFIG={ALL | CPU | IOC | CHN | CTL | DVC | ({MN | ICUU}, {addr | (r)}, {L | S})}]
	[,GLOBAL={YES | NO}]
	[,DVQ={YES | NO}]
	[,SUMMARY={ALL | ({FR | TR}, {addr | (r)})}]
	[,MF={L | (E,addr) | (E,(r)) | D[,PREFIX=p]}]

Notes:

- At least one of the operands must be specified.
- In the case of "MF=D,PREFIX=p" all further operands are ignored.

TASK= For all devices, disks and tapes of the specified task a task output record is written. The record contains information pertaining to devices, disks and tapes which have been reserved by the task.

=OWN Specifications refer to the task making the call.

=addr "addr" is the symbolic name (address) of a field (word) along with the TSN (task sequence number) for a task.

=(r) "r" is the register with the address value of "addr".

DEVICE= A device output record is written for each device specified.

{...} ALL: All devices.

FC: The entries in the subsequent list contain the code for device families (family code).

TT: The entries in the subsequent list contain the code for device types (device type code).

MN: The entries in the subsequent list contain mnemonic device names.

"addr" is the symbolic name (address) of a field which contains either a list of mnemonic device names, codes for device families or device type codes and is to be aligned on the word boundary.

Format:

- the first word (right-justified) contains the number of subsequent entries in hexadecimal notation;
- this is followed by entries, each of which is 2 bytes in length and contains either the mnemonic device name or, left-justified, the code for device families (family code) or the device type code. Surplus bytes are set to X'00'.

"r" is the register with the address value of "addr".

Example 1:

2 device types with device type codes X'A9' (=D3465) and X'91' (=FD3170)

				X'02'	X'A9'	X'00'	X'91'	X'00'	

addr↑

Example 2:

2 device families with family codes X'A0' (= disk devices) and X'90' (= floppy disks)

				X'02'	X'A0'	X'00'	X'90'	X'00'		

addr↑

DISC= For each specified disk (disk device), a disk output record is written.

{...} MONITORED: All disks for which a mount or monitoring job exists.

SCHEDULED: All disks for which explicit user specifications were made using the SET-DISC-PARAMETER command.

VSN: The entries in the subsequent list contain the vsn's (Volume Serial Number) of the disks.

MN: The entries in the subsequent list contain mnemonic device names.

"addr" is the symbolic name (address) of the field which contains a list of vsn's or mn's and is aligned on word boundary.

Format:

- The first word (right-justified) contains the number of subsequent entries in hexadecimal notation
- This is followed by entries, each of which is 8 bytes in length. They contain (left-justified) either vsn's or mn's. Surplus bytes are set to X'00'.

"r" is the register with the address value of "addr".

L: Information is output using the long format.

S: Information is output using the short format.

TASKS: Output of a list with the TSNs of the tasks currently working with the disk. The list consists of 4-byte entries and is only output for private disks for which USE=DMS has been specified.

CONFIG= A CONFIG output record is written for each specified unit of the configuration.

Note:

In the case of large configurations, the resulting output area may be extremely large when CONFIG=ALL is specified.

{...} ALL: All units of the configuration

CPU: All central processing units

IOC: All input/output processors

CHN: All channels

CTL: All multi-device controllers

DVC: All devices

MN: The entries in the subsequent list contain mnemonic device names.

ICUU: The entries in the subsequent list contain device addresses (where device address is the path definition for a device). The device address is 2 bytes in length and is structured as follows:

- 1st half-byte = number of the I/O processor
- 2nd half-byte = channel number
- 2nd byte = connection number of multi-device controller and connection number of the device (if connected to the channel directly).

"addr" is the symbolic name (address) of a field which contains a list of entries and is aligned on word boundary.

Format:

- The first word (right-justified) contains the number of entries in hexadecimal notation
- This is followed by entries, each of which is 4 bytes in length and contains, left-justified, either mn's or ICUUs. Surplus bytes are set to X'00'.

"r" is the register with the address value of "addr".

L: Information is output using the long format.

S: Information is output using the short format.

GLOBAL= This specifies that the setting of all global NDM control parameters is to be output.

=YES The output record is written.

=NO The output record is not written (default value).

DVQ= This specifies that information pertaining to the device queue is to be output.

=YES DVQ output records are written.

=NO No DVQ output records are written (default value).

Note:

The DVQ operand may only be issued by the system administrator (TSOS).

SUMMARY= A SUMMARY output record is written for each specified device family (device type). It contains information on the device family or device type. For each device family, the number of SUMMARY records of its device types is provided.

FR: The entries in the subsequent list contain the code for device families (family code).
TR: The entries in the subsequent list contain the code for device types (device type code).

"addr" is the symbolic name (address) of a field which contains a list of entries and is aligned on word boundary.

Format:

- The first word (right-justified) contains the number of entries in hexadecimal notation
- This is followed by entries, each of which is 2 bytes in length and contains, left-justified, the family code or device type code.

"r" is the register with the address value of "addr".

MF= This defines the macro format (cf. section 2.3.2).

L L-form; an operand list is generated.

(E,addr) E-form; "addr" is the symbolic name (address) of the field with the operand portion.

(E,(r)) E-form; "r" is the register with the address value of "addr".

D[,PREFIX=p] D-form; a DSECT for the operand list is generated. "p" is the character that prefixes all symbolic names occurring in the DSECT; default value: p = N.

Function 2: Generation of DSECTs for output records

Operation	Operands
DSTATUS	RECORD= HEADER / TASK / DEVICE / TAPE / DISC / CONFIG / GLOBAL / DVQ / SUMMARY / ALL

RECORD=HEADER DSECT for the output control record.
The output control record contains:

- pointers to the various output records
- counters
- length specifications (length of the output area, length of the individual output records)
- return codes for the various output records and
- equations for the global return code (register R15).

The layout is reproduced on the following page.

=TASK DSECT for the task output record (TASK operand in the macro for function 1).

=DEVICE DSECT for the device output record (DEVICE operand)

=TAPE DSECT for the tape output record (TAPE operand).

=DISC DSECT for the disk output record (DISC operand).

=CONFIG DSECT for the CONFIG output record (CONFIG operand).

=GLOBAL DSECT for the GLOBAL output record (GLOBAL operand).

=DVQ DSECT for the DVQ output record (DVQ operand).

=SUMMARY DSECT for the SUMMARY output record (SUMMARY operand).

=ALL All DSECTs (RECORD=HEADER to RECORD=SUMMARY).

For the layout of the DSECTs for the output control record (RECORD=HEADER) see the "Executive Macros" manual.

Return codes and error display

After the DSTATUS macro is called (function 1), register R1 contains the address of the output area.

The return codes for the various output records are shown in the output control records. For an explanation of the return codes see DSECT, output control record.

R15:

			a	a	a	a

A (global) return code pertaining to the execution of the DSTATUS macro is transferred in the two rightmost bytes of register R15.

Global Return code	Explanation
X'0000'	Normal execution. Information complete.
X'0004'	System error; output area not created.
X'0008'	Macro executed incompletely (output records incomplete or missing).
X'0108'	Online data for disks currently not available.
X'0208'	Online data for tapes currently not available.
X'0308'	Online data for disks and tapes currently not available.

Note:

In the case of return codes X'0108', X'0208', X'0308' repeat the call (monitoring task in wait state).

ERASE ERASE FILE (TYPE S)

The ERASE macro is used to make a file (or file generation, file generation group or temporary file) logically empty, deallocate the storage space assigned to a file and/or remove the file entry from the catalog. The macro processes partially qualified file names.

Operation	Operands
ERASE	{pathname | * | *SYSOUT | *SYSLST | *SYSLSTnn | *SYSOPT} [,{DESTROY | DATA | SPACE | CATALOG}] [,POS={A[FTER] | B[EFORE]}] [,VOLUME=vsn]

For a description of the operands see the ERASE command.

Programming notes:

- Register 15 is set to 0 upon successful completion of the macro. The error codes for unsuccessful execution are specified in the IDEMS macro (Appendix A.3).
- If a partially qualified file name is specified, and one or more of the files cannot be erased (e.g. due to ACCESS=READ), the user receives error code X'06D6' and erasure continues.
- If *SYSOUT is specified during interactive operation, return code 05DF is placed in register 15. If the EAM file to be erased (*SYSLST, *SYSOPT) does not exist or is empty, no return code is given.

FILE DEFINE FILE ATTRIBUTES (TYPE S)

The FILE macro is used to define files (or file generations or temporary files) and their attributes. With the exception of EAM files, all files must first be defined by this macro (or the FILE command). The file is thereby cataloged and the specified attributes remain known to the system. There is therefore no need to repeat these attributes in a FILE macro when the file is to be referenced again.

Principal functions of the FILE macro:

- Catalog a new file or file generation (see the "filename" operand).
- Generate an entry in the Task File Table (see the LINK operand and Fig. FILE-1 in the FILE command).
- Request devices and volumes (see the DEVICE, VOLUME and TSN operands).
- Cause console messages requesting the mounting of private volumes (see the MOUNT operand).
- Allocate and deallocate storage space on disks (see the SPACE operand).
- Allow processing of non-cataloged files or file generations on private volumes (see the STATE=FOREIGN operand).
- Provide information regarding data organization on tapes (see the LABEL, TPMARK and CODE operands).
- Define a file retention period (see the RETPD operand).
- Specify the "open mode" (see the OPEN operand).
- Specify the data structure for the various access methods:

Operands	Access method: SAM	ISAM	PAM
FCBTYPE	X	X	X
RECFORM	X	X	i
RECSIZE	X	X	i
BLKSIZE	X	X	i
SHARUPD		X	X
KEYLEN		X	i
KEYPOS		X	i
DUPEKY		X	i
LOGLEN		X	i
VALLEN		X	i
VALPROP		X	i
OVERLAP		X	i
PAD		X	i
WROUT		X	i

i: ignored but not rejected, since in certain cases ISAM or SAM files can be accessed with PAM.

- Request private devices, volumes and storage space for the file data section of ISAM files if this is to be separated from the file index section (see the DDEVICE, DVOLUME and DSPACE operands).

Operation	Operands
FILE	[{pathname / *DUMMY}][,LINK=link][,DEVICE=ddevice]
	[,VOLUME={PRIVATE / (PRIVATE,number) / vsn / (vsn,...)}]
	[,SPACE={primary / (primary[,secondary] / (firstpage,amount,ABS))}][,STATE=FOREIGN]
	[,RETPD=days][,OPEN={INPUT / OUTPUT / EXTEND / REVERSE / UPDATE / OUTIN / INOUT}][,FCBTYPE={ISAM / SAM / PAM}]
	[,RECFORM={{V / F / U} / ({V / F / U} [,{N / A / M}])}][,RECSIZE=reclength]
	[,BLKSIZE={STD / (STD,integer) / bufferlength}][,SHARUPD={NO / YES}]
	[,KEYPOS=displacement][,KEYLEN=keylength]
	[,DUPEKY=YES][,LOGLEN=flag1]
	[,VALLEN=flag2][,VALPROP={MIN / MAX}][,OVERLAP=YES]
	[,PAD=percent][,WROUT={YES / NO}][,DDEVICE=ddevice]
	[,DVOLUME={vsn / (vsn,...)}]
	[,DSPACE={primary / primary[,secondary]) / (firstpage,amount,ABS)}]
	[,WRCHK={YES / NO}]

For a description of the operands see the FILE command.

FSTAT REQUEST CATALOG INFORMATION (TYPE S)

The FSTAT macro transfers either a list of file names or a list of catalog entries (partially or in their entirety, but without file security passwords) to the output area.

Specification of a temporary file is possible.

Format 1

Operation	Operands
FSTAT	{[:catid:][$userid.][filename] / ([:catid:][$userid.][filename],len)}
	,{area / (S,area) / (reg1)} , {length / (reg2)}
	[,{FNAM / SHORT / LONG}]
	,VERSION=800
	[,PREFIX=id] [,ERR=(reg4)]
	[,A[CCESS] = {R[EAD] / W[RITE]}]
	[,BACKUP={A / B / C / D / E / (A / B / C / D / E ,...)}]
	[,CR[DATE] ={date / (date ,) / (,date) / (date,date)}]
	[,EX[DATE] = {date / (date,) / (,date) / (date,date)}]

Operation	Operands
	[,EXT[ENTS] = {value / (value,) / (,value) / (value,value)}]
	[,FCB[TYPE] = {P[AM] / S[AM] / I[SAM] / N[ONE] / (P[AM] / S[AM] / I[SAM] / N[ONE],...)}]
	[,FSIZE = {SIZE / value / (value,) / (,value) / (value,value)}]
	[,FROM = {CAT[ALOG] / (vsn,device)}]
	[,LA[DATE] = {date / (date, / (,date) / (date,date)}]
	[,PASS = {R[DPASS] / W[RPASS] / E[XPASS] / N[ONE] / (R[DPASS] / W[RPASS] / E[XPASS] / N[ONE],...)}]
	[,SAVE = {Y[ES] / N[O]}]
	[,SH[ARE] = {Y[ES] / N[O]}]
	[,SIZE = {F[REE]SIZE / value / (value,) / (,value) / (value,value)}]

Operation	Operands
	[,SORT = {N[O] / F[ILE]NAM}]
	[,STATE {NOCLOS / PCLOSE}]
	[,SUP[PORT] = {PUB[LIC] / PRD[ISC] / (PUB[LIC],PRD[ISC])}]
	[,GEN = {Y[ES] / N[O]}]
	[,TYPE = FGG]
	[,VTOC = {Y[ES] / N[O]}]

catid — This operand denotes the identifier of the catalog from which the file or files are to be selected.
Default value: default catalog ID of the appropriate user ID.

userid — This operand specifies a user identification. "$userid" designates all the user's files. If another user's identification is specified, only information concerning shareable files (SHARE=YES) is output.
Default value: the user's own user ID, i.e. the one used in the LOGON command.

filename — This operand specifies the file(s) for which information on the status is to be provided. This may be a partially qualified or fully qualified file name, the name of a file generation, or the partially or fully qualified name of a file generation group. The maximum length of the file name is 80 characters. Temporary files are supported; only the files of the user's own task are taken into account.

len — For the "catid", "userid" and "filename" operands a field of the length "len" is generated; this field contains the file name (left-justified) and is padded with blanks.
If this operand is not specified, a field with the required length is reserved.
Both the "catid" and "filename" operands can be used in accordance with the patterns that have been defined (see the FSTATUS command).

area — This specifies the symbolic address of the output area to which the output list is to be transferred.

(S,area) — This operand specifies the symbolic address of the output area; this is not an absolute address but may be addressed via a base register (e.g. in a dynamic main memory with a DSECT overlay).

(reg1) — This specifies the register containing the address of the output area.

length — This operand specifies the output area length.
Minimum length: 9
Default value: 60 bytes when SHORT is specified, 2048 bytes when LONG is specified.
If the specified length is insufficient, no information is transferred.

(reg2) — This is the register containing the output area length.

FNAM — A list of file names is to be transferred to the ouptut area.

SHORT — The static portion of the catalog entry (60 bytes) and the file name (maximum of 54 bytes) are transferred to the output area. These parts of the catalog entry can be given a symbolic name with the IDCE macro.

LONG — The complete catalog entry is output (the bytes for the file passwords are set to binary 0; for exceptions see "PASSWORD").

The catalog entry is made up of the following parts:

1. Static part
2. File name
3. Extension
4.a Volume table, vsn table (only for catalog entries which define files)
4.b FGG option (only for group entries for file generation groups)

The following macros exist for the provision of symbolic names:

1. IDCE (catalog entry)
2. None
3. IDCEX (catalog entry extension)
4.a IDVT, IDEE (volume table, extent list)
4.b IDCEG (catalog option for FGG)

PREFIX=id — Identifier for distinguishing between several calls of the FSTAT macro in one program section. One to three characters are permitted for the identifier, the first of which must be a letter.

ERR=(reg4) — Register for accommodating additional error information (besides register 15).

VERSION=800 — The macro for Version 8.0 is called. If this operand is omitted, the macro for the earlier version is called (see format 2).

Note:

For "reg1", "reg2" and "reg4" the values 1 and 15 are prohibited, while 1 must not be specified for "reg3".
The specifications must be pairwise disjoint (AND/OR logic).

For a description of the remaining operands see the /FSTATUS command.

Specification of the output area

The following paragraphs describe the output format generated by the FSTAT macro.

Information concerning files in various catalogs or under various user IDs can be transferred to the output area with the FSTAT macro by specifying patterns in the PVSID or in the user ID. This information is not contained in the catalog entry and must therefore be additionally transferred to the output area.

In this new version of the FSTAT macro, the structure of the output area is as follows:

ANZ	LEN4	PREFIX	Data	
	LEN4	PREFIX	Data	
		.		
		.		
	LEN4	PREFIX	Data	EOL4	RET

where

ANZ: is the total number of files affected by the FSTAT macro (4 bytes)

LEN4: is the length of the subsequent PREFIX and subsequent data + 4 (4 bytes)

PREFIX:

LP	LC	catid	LU	userid

LP: Length of PREFIX (2 bytes)
LC: Length of "catid" + 2 (2 bytes)
catid: Catalog identifier for all entries in the subsequent data section
LU: Length of the "userid" + 2 (2 bytes)
userid: User ID for all entries in the subsequent data section

EOL4: End-of-list identifier (X'00000000')

RET: X'00' All requested information is transferred to the output area.
X'01' Only incomplete information is transferred to the output area after a request.

Note:

If no files have been affected by the macro, the output area is structured as follows:

ANZ	EOL4	RET

(Length: 9 bytes)

In turn, each "data section" comprises a list. The "user information" fields within this list contain the information which the user has actually requested.

LEN2	user information	LEN2	user information ....

..... user information	EOL2

where

LEN2: Length of the subsequent user information + 2 (2 bytes)

EOL2: End-of-list identifier (X'0000')

User information is dependent upon the input operand FNAM, SHORT or LONG and is composed of:

FNAM: File name

The maximum length of this user information is 41 bytes.

SHORT:

static portion CE	file name

The length of the static portion of the catalog entry is 60 bytes. The last byte contains the length of the file name which follows.

The maximum length of this user information is 101 bytes.

LONG: Complete catalog entry (cf. section 1.2)

The maximum length of this user information is 2032 bytes.

Error recovery

Return codes are stored in register 15 and in a second register (ERR operand) for purposes of error recovery.

The following return codes are transferred in register 15:

X'00000000' Request successfully completed.

otherwise DMS error code in the event of unsuccessful completion (see the IDEMS macro).

In addition, the following return codes are stored in the register specified by the ERR operand.

Byte3	Byte2	Byte1	Byte0

Byte0: X'00' Required information transferred completely to output area.
X'01' Only part of required information transferred to output area.

Byte1: More exact specification for DMS error "06B8" in register 15

X'00' DMS error "06B8" did not occur
X'04' Invalid operand
X'08' Error in the GEN operand
X'0C' Error in the TYPE operand
X'10' Error in the STATE operand
X'14' Error in the VTOC operand
X'18' Error in the SUPPORT operand
X'1C' Error in the VOLUME operand
X'20' Error in the FCBTYPE operand
X'24' Error in the ACCESS operand
X'28' Error in the SHARE operand
X'2C' Error in the PASS operand
X'30' Error in the BACKUP operand
X'34' Error in the xxDATE operand (CRDATE,EXDATE,LADATE)
X'38' Not used at present
X'3C' Not used at present
X'40' Error in the FSIZE operand
X'44' Error in the SIZE operand
X'48' Error in the SAVE operand
X'4C' Error in the EXTENTS operand
X'50' Error in the FROM operand
X'54' Error in the SORT operand
X'58' FROM operand together with VOLUME, SUPPORT on VTOC operand

Byte2 and byte3: Not used at present (constant X'00')

Format 2

Operation	Operands
FSTAT	[pathname],area,[length][,{SHORT | LONG}] [,GEN={YES | NO}][,TYPE=FGG][,STATE=NOCLOS] [,SUPPORT={PUBLIC | PRDISC}][,VOLUME=vsn][,VTOC={Y[ES] | N[O]}]

"pathname" stands for: {:catid:[$userid.][filename] | $userid.[filename] | filename}

catid — This operand specifies a catalog identifier.

userid — This operand specifies a user ID. The specification "$userid." refers to all files belonging to a user. If the ID of any other user is specified, only information on shareable files (SHARE=YES) is output.
Default value: the user's own ID.

filename — This operand specifies a fully or partially qualified file name. In the following instances merely a list of file names is transferred to the user area:

- when a partially qualified file name is specified;
- when a fully qualified name for a file generation group is specified in conjunction with GEN-YES.

area — The symbolic address of the user area to which the catalog entry or the list of names is to be transferred.

length — This specifies the length of the user area.

Default values:

60 bytes when SHORT is specified, and
2048 bytes when LONG is specified.

If the specified length is insufficient, no information is transferred.

SHORT — The static portion of the catalog entry (60 bytes) is transferred to the user area (see "filename" for exceptions). This part of the catalog entry can be given a symbolic name using the IDCE macro.

LONG

The complete catalog entry is output (the bytes provided for passwords are set to binary 0).

The catalog entry consists of the following parts:

1. Static part
2. File name
3. Extension
4.a Volume table, vsn table (only for catalog entries which define files)
4.b FGG option (only for group entries for file generation groups)

The following macros exist for the provision of symbolic names:

1 IDCE
2 None
3 IDCEX
4.a IDVT,IDEE
4.b IDCEG

GEN=YES

If the specified file name refers to one (or more) file generation groups, this operand results in output of the group name, followed by all generation names. These names are not sorted.
Each name is preceded by a 1-byte length field containing the value "name length + 1".

=NO

No information concerning file generations is requested.

TYPE=FGG
STATE=NOCLOS
SUPPORT=PUB[LIC]
SUPPORT=PRD[ISC]

These operands are meaningful only if the specified file name is partially qualified or refers to a file generation group. Specification of one of these operands results in a selection being made from all catalog entries referenced by the file name, followed by output of the associated names (cf. Notes).

VOLUME=vsn

This operand outputs all file names or catalog entries of files which occupy space on the specified volume.

VTOC=Y[ES]

This operand outputs the VTOC catalog entries (from the F1 label of a private disk) on the basis of the last current status in the entire computer network.
This entry causes all other keyword operands to be ignored.

=[NO]

This operand outputs the TSOS catalog entries on the basis of the last current status in an individual co-user. (Co-user: CPU for which a random access device was defined as shareable.)

Programming notes:

- Selection of the catalog entries takes place when the following operands are specified:

  TYPE=FGG

  Selection is on the basis of the catalog entries of file generation groups. However, the names of the generations of an FGG are only output if GEN=YES is specified as well.

STATE=NOCLOS Selection is on the basis of the catalog entries of files which are not closed. In this case selection also covers the generations of an FGG. GEN=YES need not be specified.

SUPPORT= Selection is on the basis of the catalog entries of files and file generation groups on

- public volumes (=PUBLIC)
- private disks (=PRDISC)

- If a combination of the TYPE, STATE and SUPPORT operands is specified, the catalog entries satisfying all of the conditions are selected.
- If a fully qualified file name is specified, the catalog entry (SHORT or LONG) for this file name is transferred to the specified area. The password fields are set to binary zero:
- A name list with the following format is output:

Length of the file name + 1 (1 byte)	File name (up to 41 bytes)

...

End of list (1 byte, X'00')	Output complete (1 byte)? yes (X'00'), no (X'01')

...

The name of a group entry is followed by the character string "␣(FGG)". The length field has the value: length of the group name + 7.

The list ends with a byte containing the value X'00'.

A further byte indicates whether the user area was large enough to accommodate all the file names.

where:

X'00' All the file names have been transferred to the user area.

X'01' One or more file names could not be transferred to the user area as it was already full.

- The following return codes are set in register 15:

00 The request was successfully completed.

The error code for unsuccessful termination is defined in the IDEMS macro.

Example:

06 KLAUS	0B ABRECHNUNG	00	00	(ABRECHNUNG = payroll)

IMPORT CREATE CATALOG ENTRY FOR PRIVATE FILES (TYPE S)

The IMPORT macro catalogs files contained on a private disk (see the IMPORT command).

Operation	Operands
IMPORT	[pathname],VOLUME=vsn,DEVICE=device ,AREA=(addr,length)[,REPLACE={YES | ABS | NO}] [,GEN={YES | NO}][,LIST=({YES | NO | ONLY})] [,PVSID=catid]

AREA=(addr,length)
"addr" is the symbolic address of a user area to which the output information is to be transferred. "length" specifies the length of the user area. If the operand LIST=NO is specified, this entry can be omitted.

For a description of the remaining operands see the IMPORT command.

Programming notes:

- Each item of information transferred to the user area is 56 bytes long and has the following format:

pathname (54 bytes)	Coded return information (2 bytes)

- The coded return information has the following meaning:

  Leftmost byte: This byte is significant only if the macro is used by the system administrator.

  Rightmost byte:

  C'0' The file has been newly cataloged.

  C'1' A file already cataloged with this name has been erased.

  If the operand LIST=ONLY was specified, this value simply means that a file with this name already exists. The protection attributes are not checked in this case.

  C'2' A file with this name already exists and the operand REPLACE has the value NO.

  C'3' The already cataloged file is protected (error when erasing this file).

C'4'	System error when reading the catalog.
C'5'	A catalog entry for this file on the private disk already exists.
C'6'	System error when reading the F1 label of the private disk.
C'7'	The file is a file generation whose absolute generation number is not compatible with that which already exists (generation number too high).
C'8'	Catalog entry already exists and was overwritten.
C'9'	Catalog entry already exists and the file is locked.

If the value of the return information is equal to C'0', C'1', C'5' or C'8', this indicates that a file on private disk has been successfully processed.

RDTFT OBTAIN INFORMATION FROM TFT AND TST (TYPE S)

By means of the RDTFT macro, the user can cause status information from the TFT relating to the files and devices currently in use to be output to a user area. Furthermore, he can obtain information from the associated TST.

Operation	Operands
RDTFT	area ,[length] [,{SHORT / LONG}] [,FILE=pathname] [,LINK=link]

area — This operand specifies the symbolic address of the user area to which the information from the TFT is to be transferred.

Specification of the operand "area" is mandatory.

length — This operand specifies the length of the user area.

Default value: 140 bytes if SHORT is specified,
2048 bytes if LONG is specified.

SHORT — The static portion of the Task File Table, followed by the static portion of the associated TST, are transferred to the user area (i.e. without the device table).

LONG — The complete Task File Table (TFT), followed by the associated TST including the device table, are transferred to the user area.

By means of the DMADR macro, the information output in response to LONG or SHORT can be provided with symbolic names, provided that the LINK operand was specified.

Note:

Positional operands: a comma must be entered for "length" if SHORT or LONG is specified, but "length" is not.

FILE=pathname — Only the file name and the associated link name are transferred to the user area.
In this case, the DMADR macro cannot be used for symbolic addressing.

"pathname" stands for: [:catid:][$userid.]filename

catid — This operand denotes the catalog identifier associated with the file. If this entry is missing, the default catalog ID assigned to the user ID is assumed.

userid — This operand denotes the user identification associated with the file. If this operand is omitted, the user's own user ID is assumed.

filename — This operand specifies the fully or partially qualified file name (up to 41 characters). If the file name specified does not begin with the user ID, it is prefixed by the user ID of the job.
If "filename" designates a file generation of a group, the absolute value of the generation number must be specified.
When temporary files are specified, the internal file name is output.

LINK=link — This operand specifies the file link name.
If the LINK operand is specified, only the information from the TFT (including the TST linked with it) is transferred to the area. Otherwise the selection of the TFT is not dependent on the file link name.

Notes:

- No TST information can be transferred to the user area unless the Task File Table (TFT) is linked with a TSET. If it is not, the appropriate section within the user area is set to binary zero.

- If the LINK operand has been specified, only status information relating to the TFT entries associated with the file link name is transferred to the user area.

- Output takes the following form, as indicated in the DMADR macro, providing the LINK operand has been specified:

Length of the path name + 8 bytes + length of the TST + 2 (2 bytes)	File link name (8 bytes)	Path name (54 bytes)	TST static portion	TST Volume information for the LONG operand

- If the LINK operand was not used, or if a file link name consisting of blanks was specified, a chronological list of file link names and the file names associated with them is output, i.e. the file link name of the TFT created last is transferred first.

- The list is terminated by a byte containing the value X'00'. A further byte indicates whether the user area was large enough to accommodate all the file link names and file names.

  Where:

  X'00' All the file link names and their file names have been transferred to the user area.

  X'01' One or more file link names, together with their file names, could not be transferred because the user area was already full.

- The following return codes are set in register 15:

  X'00' - request successfully completed.

  The error codes for unsuccessful execution are specified in the IDEMS macro (Appendix A.3).

REL DELETE TFT ENTRY (TYPE S)

The REL macro deletes an entry in the Task File Table.

Operation	Operands
REL	[Link]

For a description of the operands see the RELEASE command.

Programming note:

Register 15 is set to zero upon successful completion of the macro. The error codes for unsuccessful execution are specified in the IDEMS macro (Appendix A.3).

VERIF RESTORE FILE (TYPE S)

The VERIF macro serves to restore corrupted files (see also section 2.1.7, "Restoration of Corrupted Files").

Operation	Operands
VERIF	pathname1[,[pathname2][,REPAIR={YES | NO | ABS}]]

"pathname1" stands for: [:catid:][$userid]filename1
"pathname2" stands for: [:catid:][$userid]filename2

filename1 Fully qualified name of the file to be restored.

For a description of the remaining operands see the VERIFY command.

Programming note:

Register 15 is set to 0 upon successful completion of the macro. The error codes for unsuccessful execution are specified in the IDEMS macro (Appendix A.3).

5 ACCESS TO FILES

Each program that has to process a file must perform the following actions when it executes in a job:

- Set up the file (using a FILE macro).

  This action is not required if the file is already cataloged and has been provided with storage space.

- Open the file (using an OPEN macro).

- The system checks whether the calling job is authorized to access the file and prepares for the actual access to the contents of the file.

- Access the file contents (using access methods of the DMS).

  Upon request the system transfers data between the calling job and the file.

- Close the file (using a CLOSE macro).

  The system logically disconnects the calling job from the file, after which data transfer requests are rejected.

A job can open a file, access its contents and close it as often as it likes.

It should be noted, however, that opening and closing a file are very time-consuming actions; consequently, a job should not open a file until it is able to process it, and close it only once processing is complete.

5.1 OPENING FILES

5.1.1 General

The OPEN macro serves to open a file. This macro associates the file to be opened with a File Control Block (FCB).

The File Control Block therefore contains a file link name, and optionally the file name. It also describes the file's technical characteristics, as seen by the program.

What information is required by the DMS for opening a file?

The following points need to be clarified before the DMS is able to open a file:

- Which file is to be opened?
- Does the calling job satisfy the conditions stipulated by the file's protection attributes?
- With which access method is the file to be processed? How is the file organized (block size, record length etc.)?
- Is the file to be opened as an input file or an output file? If it is to be opened as an output file, is it to be replaced or to receive further data?
- In which part of the address space are data buffers to be set up? In user memory (class 6) or in system memory (class 5)?
- How are the logical records to be transferred to the job and received by the job? Is in each case the address at which the current logical record begins in the buffer to be transferred (locate mode), or is the record in a user area to be transferred (move mode)?
- Are other jobs permitted to access this file simultaneously (shared file update)?
- If the file is on private volumes or is to be created on private volumes: Which volumes should be used?
- At which address is the program run to be continued in the event of an error or a contingency?

From where does the DMS obtain the information required?

The DMS obtains the answers to the above questions from the following sources:

1. A FILE command containing the same file link name as the program's File Control Block. (These details are kept internally in the Task File Table.)
2. The program's File Control Block, to which the OPEN macro refers.
3. The catalog entry for the specified file. (This means a file must always be cataloged before it can be opened.)

4. When checking the access authorization, the DMS uses additional information originating from a PASSWORD command and which is stored internally in the job's password table or in the File Control Block (field ID1PASS).

5. To determine the open mode (input file, output file), the DMS uses, in addition to 1. and 2., an entry in the OPEN macro. (The entry in the OPEN macro has priority over those in 1. and 2.)

The following applies to information contained in more than one of the sources 1. through 3.:

- Information in the FILE command has priority over information in the File Control Block and the catalog entry.
- Information in the File Control Block has priority over information in the catalog entry.
- If information from 1. and 2. is incompatible with the contents of the catalog entry to the extent that the file cannot be processed, the DMS aborts opening with an error code.

Exception:

The DMS always takes information concerning the retention period of an existing file directly from the catalog entry.

5.1.2 File Control Block (FCB)

An FCB can be created/modified in the following stages:

1. Using the FCB macro when the program is written. This macro reserves storage space for the control block and optionally supplies information to it.

2. During program execution, prior to opening the file, by directly writing in current values. The IDFCB macro may be used to do this (see Appendix A.2).

3. While the file is being opened, the user is able to check and, if necessary, modify the File Control Block after certain processing steps of the OPEN routine (see the EXLST macro, operands OPENX and OPENZ).
   Thereafter modifications to the FCB by the program are not permitted.

Functions of the File Control Block during file processing

As long as a file remains open, the File Control Block contains all the information describing the current status of the file.

This information is usually of significance to the user only after an error exit has been executed (see the FCB macro, operand EXIT= and the EXLST macro).

In this case, the File Control Block also contains a DMS error code and an identifier for the error exit (see also section 5.3, "Handling Errors and Contingencies").

5.1.3 Sequence of Events when Opening a File

The DMS carries out the following activities when a disk file is opened:

- Complete the File Control Block from the FILE command and catalog entry sources.
- Allocate buffer areas.
- Request volumes (for files on private volumes).
- Load access method routines; these routines carry out the blocking and deblocking of logical records and are treated as a part of the user program.
- Set up a P2 File Control Block (P2FCB); this is a control block used internally by the DMS and which cannot be accessed by the user.
- Update the catalog entry (for output files).

A detailed description of the sequence of events involved in opening disk files can be found in the following section.

Detailed sequence of events in opening a file

1. Search the TFT (Task File Table) entries for the current file link name (contained at this point in the File Control Block, and which either originates from an FCB macro or was written directly into the control block).

   If the link name is not found, a TFT entry is created. If the link name is found, the FCB of the user is updated due to the TFT information, as the TFT contains FCB information provided by a previous FILE command.

   Notes:

   - If the LINK operand in the FCB has the value 8X'40' (8C'␣'), the DMS reserves a new TFT entry without checking whether there already is a TFT entry with the value specified. This new TFT entry is marked as "active". The following fields are then transferred from the FCB to the TFT.

     file link name,
     file name,
     OPEN mode.

   - If there is no TFT entry with the specified LINK value, a new TFT entry is requested, marked as active, and the same fields are transferred to the TFT.
     At the time the CLOSE command is issued, such an entry is automatically released by the system.

   If a TFT entry is found, and this entry was created by a FILE command, the following fields are transferred from the TFT to the File Control Block:

SHARUPD	*KEYLEN	*VALPROP
filename	PAD	*LOGLEN
RETPD	*RECFORM	*VALLEN
*RECSIZE	*FCBTYPE	WROUT
*BLKSIZE	DUPEKY	OPENmode
*KEYPOS	OVERLAP	

Notes:

- Only those fields specified in a previous FILE command are transferred.

- Fields of the TFT entry which have the value 0 (i.e. the corresponding operand in the FILE command was specified as a null operand) are not transferred to the File Control Block; however, a bit is set in the corresponding field of the File Control Block. These fields are completed later from catalog information (applicable only to the fields marked with an asterisk in the above list). See also 2. below.

2. Seek and read the catalog entry for the file. Complete the undefined fields in the File Control Block using information from the catalog, if possible.

   Only those fields in the catalog entry which have null values are transferred to the File Control Block (see fields marked with an asterisk in 1. above).
   The way in which the catalog is searched depends on how the file name was specified:

   - If the file name contains a user identifier ($userid.), this user ID is used for the search. If no file is found under "$userid.", an error message with error code OD33 is issued.

   - If the file name is specified without any user ID, searching is performed using the user ID under which the job is executing. If no file is found under this user ID, the open formats differ, as described below:

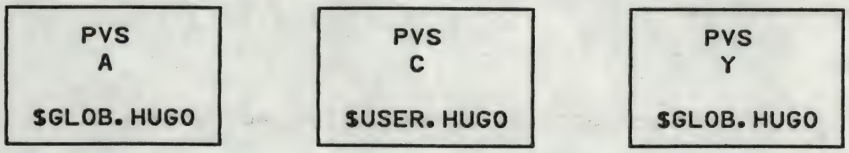
     1. Standard case: Error message DMSOD33 is output.

     2. If the operand OPTION=GLODEF was specified in the FCB macro, the file is searched for under the ID which the system administrator stored in the class 2 option DEFLUID. If the file cannot be found under this ID either, the error message is likewise OD33.

   Example:

   Defining
   DEFLUID :Y:$GLOB

   The job $USER identification
   Standard catalog identifier C
   Option=GLODEF set



Opening files

Specification of	is the equivalent of	Result
HUGO	:C:$USER.HUGO	File found under default catalog ID
$HUGO	:Y:$GLOB.HUGO	Catalog ID Y and user ID $GLOB are completed from the DEFLUID specification. The file is found on PVS Y.
:C:HUGO	:C:$USER.HUGO	File found on PVS C.
:A:HUGO	:A:$GLOB.HUGO	Only the user ID $GLOB is completed. The file is found on PVS A.

3. If the volume list for the TFT entry does not exist, it is created using information from the catalog, and the volumes are requested.
   If the volume list for the TFT entry already exists, i.e. if it was created by a FILE command, but MOUNT=0 was specified, then the volumes are requested.
   Otherwise (MOUNT=0) a check is made as to whether the correct volumes were made available.

4. In the event of contingencies, a branch is made to the OPENX exit.

5. If the OPEN operand has the value OUTPUT or OUTIN, the following fields are transferred from the File Control Block to the catalog entry:

   FCBTYPE
   BLKSIZE

   If FCBTYPE ≠ PAM:

   RECFORM
   RECSIZE
   KEYLEN
   KEYPOS
   VALPROP
   VALLEN
   LOGLEN

   In the case of OPEN INPUT, IOAREA1 and IOAREA2 are already provided with data.

6. If the OPEN operand has the value OUTPUT or OUTIN, the program receives control at the OPENZ exit, assuming this exit is provided for in the program. Modifications now made to the File Control Block no longer affect the catalog entry.

7. Verify passwords.

8. Allocate input/output buffers and validate buffer addresses, if necessary.

9. Establish connection with the access method. Load routines for blocking and deblocking logical records (SAM, ISAM).

10. Update the catalog entry if the file was opened in OUTPUT or OUTIN mode.

Warning:

Technically speaking, a file does not exist until it has been opened once in OUTPUT or OUTIN mode, and then closed again.

5.2 CLOSING FILES

Any file that has been opened must be closed with a CLOSE macro after processing. This may be done in the following ways:

- Closing one file.
  The CLOSE macro designates the file to be closed via the File Control Block that was also used for opening the file.

- Closing all the files that have been opened by the calling job at this time.

The DMS carries out the following activities when a disk file is closed:

- Release all the work areas occupied in class-5 memory during file processing (e.g. buffers allocated automatically when the file was opened).

- Update the catalog entry if necessary (for output files).

At this point the following information is placed in the catalog entry: creation date, expiration date, and a pointer to the last PAM block of the file.

- Wait for completion of all outstanding write operations.

- Unlock all PAM blocks still locked in the file for the calling job.

- Provide the FCB with the contents it had before it was opened.

5.3 HANDLING ERRORS AND CONTINGENCIES

During the processing of a file, situations may arise that require a decision from the task, e.g. if end of file is reached during reading, or if an error occurs, or if conflicts with other jobs arise during shared file update.

To enable the DMS to communicate with the user job in such situations, an exit address should be specified in the File Control Block (operand EXIT= in the FCB macro).
If this address is not available, the DMS aborts program execution if errors or contingencies occur.

The following options are available to the programmer for defining exit addresses for different classes of contingencies and errors:

1. Specifying, in the operand EXIT= of the FCB macro, the address of a program section which is responsible for processing all errors which may occur. This address must be enclosed in parentheses, i.e. EXIT=(addr).

   The ID1XITB field of the File Control Block should then be examined in this program section to determine which event caused the exit.

   Depending on the type of event, it may also be necessary to examine the ID1ECB field to obtain an error code.

   See Appendix A.3 for further details.

2. Specifying the address of an EXLST macro in the operand EXIT= of the FCB macro. This address should be specified without parentheses, i.e. EXIT=addr.

   For each type of event, the address of a program section responsible for processing this event should then be given in the EXLST macro.

   Processing continues as described under 1.; field ID1XITB need not be examined, however, if an event type corresponds exactly to the appropriate EXLST exit. (This applies to all EXLST exits except COMMON=.)

3. Using technique 2. and specifying in the operand COMMON= of the EXLST macro the address of a program section responsible for several types of events. This program section should be constructed as described in 1.

No matter which option is used, the following conventions apply:

- The DMS places the address of the File Control Block in general register 1 before passing control to the user program. (In the case of multiple files, therefore, it is possible to use the same program sections for handling errors.)

- The DMS transfers to the ID1RTNAD field of the File Control Block the address of the instruction which, during normal processing, would be processed next. This address refers either to the next instruction in the user program or, in the case of some action macros (e.g. PUT, GET), to the File Control Block.

- The type of event is contained in coded form in the ID1XITB field of the File Control Block. The various codes are listed in Table 5-1. (See the EXLST macro.)

Errors and contingencies

- In the case of the EXLST exits ERROPT, OPENX, OPENZ and WLRERR the program cannot be continued until the EXRTN macro is specified.
- When a special status occurs during file processing, fields ID1XTB and ID1ECB in the associated P1FCB are provided with the appropriate information by the system. These fields are not reset when processing is continued by the DMS, i.e. they are not updated until another special status occurs.

Note:

The DMS error codes are listed using the IDEMS macro.

However, the texts describing the various errors are very brief. More detailed information can be obtained using the HELP command; here the leading zero of the DMS error code should be replaced by DMS0.

In the same way the meanings of the error messages and the possible measures ito be taken errors occur can be found in the "System Messages" manual.

5.4 SERVICE MACROS FOR DISK FILES

CLOSE CLOSE FILE (TYPE R)

The CLOSE macro is used to disconnect a file from a user program. All I/O buffers previously automatically secured by the system are also released. The FCB is reset to the contents it had before the OPEN macro was issued.

Operation	Operands
CLOSE	{ ALL / fcbaddr / (1) }

ALL — This operand specifies that all files for the job, except system files (e.g. SYSLST), are to be closed. User EAM files are not closed. If a file is closed abnormally, a warning message is issued.

fcbaddr — This operand specifies the address of the FCB associated with the file to be closed (exit address).

(1) — Register 1 contains the FCB address.

EXLST CREATE BRANCH ADDRESS LIST ‹TYPE 0›

The EXLST macro is used to convey to the DMS the addresses of program sections for handling errors and contingencies.
The values of the individual operands are symbolic addresses of the associated program sections. NO indicates that the program does not wish to receive control if the relevant condition occurs.
The default function of each operand is underlined. To nullify the default value, it is necessary to specify the keyword with a null operand.

Format for disk files

Operation	Operands
EXLST	[CLOSER={relexp | NO}] [,COMMON=relexp] [,DLOCK={relexp | NO}]
	[,DUPEKY={relexp | NO}] [,EOFADDR={relexp | NO}] [,ERRADDR={relexp | NO}]
	[,ERROPT={relexp | IGNORE | SKIP | NO}] [,ISPERR={relexp | NO}] [,LOCK={relexp | NO}]
	[,NODEV={relexp | NO}] [,NOFIND={relexp | NO}] [,NOSPACE={relexp | NO}]
	[,OPENC={relexp | NO}] [,OPENER={relexp | NO}] [,OPENX={relexp | NO}]
	[,OPENZ={relexp | NO}] [,PASSER={relexp | NO}] [,PGLOCK={relexp | NO}]
	[,SEQCHK={relexp | NO}] [,USERERR={relexp | NO}] [,WLRERR={relexp | NO}]

relexp — This stands for "relative expression"; relexp is normally a symbolic address in the Assembler program.

CLOSER= — An error was encountered while attempting to CLOSE the file (for example, when attempting to write labels). An error code detailing the condition is stored in the FCB.

COMMON= Control passes to this address if there is no entry for the type of error encountered.

Exceptions:

ERROPT
OPENC
OPENX
OPENZ

See Table 5-1, "Use of error exits" at the end of the EXLST description.

DLOCK= This exit applies only to UPAM.

Control is passed to this exit if another job has set one or more locks necessary to complete the current PAM action macro, and there is danger of a deadlock situation.

By means of the FCB parameter PAMTOUT, the user can define the number of seconds he wishes to wait to obtain a lock before this exit is taken.

The danger of deadlock exists whenever a user wants to obtain further locks but not all of them are available, while he has already set one or more locks himself.

If he is given control at this exit, any attempt to obtain further locks before all currently held locks are released will result in abnormal termination of the program.

Under ISAM, all current locks must be released before an attempt is made to obtain another set of locks. This means that no potential deadlock situation can arise in ISAM. Accordingly, control is never passed to this exit during execution of an ISAM action macro.

DUPEKY= If the operand DUPEKY=YES is not specified in the FCB macro and duplicate keys occur, control is passed to the address which has been specified.

The INSRT macro can be used to insert a record in a file when the key of this record already exists (independent of DUPEKY=YES).

EOFADDR= The end-of-file condition has been encountered with a read attempt.

ERRADDR= A hardware malfunction or an abnormal termination has occurred during processing of the file. Status bytes consisting of the standard device byte, the Executive flag byte and the three sense bytes are stored in the FCB.

Note:

SAM read errors use other exits (EOFADDR, ERROPT, USERERR).

ERROPT= This entry applies to SAM input files, and specifies functions to be performed in the event of an errored block.

If a parity error is detected when a block of records is read, the block is reread a standard number of times before it is considered an errored block. After this, the job is automatically terminated, unless the ERROPT entry specifies a particular reaction in the event of an error. Either IGNORE, SKIP or the symbolic name of an error routine can be specified.

The functions of these three specifications are:

=relexp The DMS branches to the user routine that performs any desired function, i.e. processes the records or takes appropriate measures.
General register 0 contains the address of the errored block.

In his error routine, the programmer must not issue any GET macros for records in the errored block. In this routine, the user can issue logical macros (GET, PUT) to any file other than the file containing the errored records. If the program wishes to return to normal processing, the EXRTN macro must be issued. If general register 0 contains X'00000001', the current block will be skipped and processing continues with the next block. Any other code indicates that this block is to be processed as if no error had occurred.

=IGNORE The error condition is completely ignored, and the records are made available to the user for further processing.

=SKIP No records in the errored block are made available for further processing.
The next block is read and processing continues with the first record of that block.

This entry does not apply to output files. It applies to records with invalid lengths if the WLRERR name entry is not included.

ISPERR= Not enough space is available to extend the index of an ISAM file. With ISAM files, space may be extended either for the index or for the data, even though it is the same file.

LOCK= The condition occurred because the program attempted to open a file in a mode other than INPUT and it was already open for another user. Similarly, if the file is currently open for another user in a mode other than INPUT and the user tries to open the file as INPUT, the same condition occurs. This exit should be used only if another section of the program can be executed in the event of this condition occurring.

NODEV= No free device exists upon which the private volume may be mounted; or, the private volume is currently being used by another user. If the SECURE, HOLD and DROP commands are used properly, this condition should not arise.

NOFIND= A record with the defined key could not be found when the GETKY macro was issued, or a record which matched the GETFL criteria could not be found within the specified range.

NOSPACE= Insufficient space exists with which to perform secondary allocation or extended output.

OPENC= This exit is taken if the file to be opened was not closed during previous processing. Restoration may be attempted using the VERIF macro (see section 2.1.7, "Restoration of Corrupted Files").

If the OPENC operand was explicitly specified in the EXLST macro, control is passed to the user at the specified address and error code X'0DD1' is issued in field ID1ECD of the FCB.
If the user has not specified the OPENC operand, the system continues OPEN processing.
If an attempt is made during processing to open a file which is still open, the user obtains error code X'0DD9' (file locked). If the file is a dummy file and the OPEN mode is unequal to OUTPUT/OUTIN, the user receives error code X'09DA'.

OPENER= An error was encountered while opening the file (for example, inconsistent FCB, no space allocated for the file). An error code detailing the condition is stored in the FCB.

OPENX= The FCB has been modified by the information supplied either in a FILE command (macro) or via the catalog. The program can now ensure that all parameters are consistent and that the OPEN macro can be completed without errors. If a new file is being created, the FCB is modified by the FILE command, although the catalog has not yet been written back to the catalog.
The program cannot be continued until the EXRTN macro is specified.

OPENZ= For a file being opened in either OUTPUT or OUTIN mode, the catalog processing has been completed by the time the exit is taken; however, the remainder of the OPEN processing must still be completed. At this stage the program can modify the FCB so as to enable further processing. For example, the catalog might indicate that the file involved is a SAM file, while the user might want to process it using PAM. The program cannot be continued until the EXRTN macro is specified.

PASSER= An invalid password was specified for a protected file.

PGLOCK= Control is passed to this exit if another job has set one or more locks necessary to complete the current PAM or ISAM action macro and there is no danger of a deadlock situation.

For PAM, the user may use the PAMTOUT operand of the FCB to define the number of seconds he wishes to wait in order to obtain a lock before this exit is taken.

For ISAM, this exit is always taken immediately if the locks requested are not available; i.e the PAMTOUT operand is ignored by ISAM.

The following entries apply to ISAM only.

If this exit is not taken in the case of SHARUPD=YES (i.e. if the default value or NO is specified), and the requested block is locked, the user has to wait until he is given access to this block. The job is automatically placed in the bourse according to its priority. The user is not informed of the "busy" status.

If a file was opened with SHARUPD=YES, the use of all ISAM macros may have the effect that control is passed to this exit (with the exception of OSTAT). If the PGLOCK exit is taken, the "internal pointer" is wrong unless the condition is due to a PUTX or ELIM macro (without KEY).

That is why it is mandatory for this "internal pointer" to be repositioned before issuing a macro which assumes it is correctly positioned (e.g. with GET, GETR and GETFL). The pointer can be repositioned by using the RETRY macro or any of the following ISAM action macros: GETKY, SETL, PUT, STORE, INSRT or ELIM (with KEY). If GET, GETR, or GETFL is issued before the pointer is repositioned, control is passed to the USERERR exit.

If the condition which caused the PGLOCK exit to be taken was due to a PUTX oder ELIM macro (without KEY), the data block remains locked and repositioning is not necessary.

SEQCHK= A record to be added to an ISAM file via the PUT macro has a key less than the highest key already present in the ISAM file.

USERERR= The program has attempted to execute a macro in an illogical manner or has called an illegal action, e.g. it has issued a WRITE macro to a file opened in INPUT mode, or it contains an invalid operand code in PAM.

The user is given control at this exit if the file is opened with SHARUPD=YES and he issues a PUTX or ELIM macro (without KEY) without first locking the data block, or if he issues a GET, GETR or GETFL macro after taking the PGLOCK exit without subsequently repositioning his file.

WLRERR= A record with an invalid length was read. For fixed-length blocked records, record length is considered invalid if the block length is not a multiple of the logical record length (specified in the FCB entry RECSIZE) and less than or equal to the maximum block length (specified in the FCB entry BLKSIZE). This permits short blocks of logical records to be read without "invalid record length" being issued. For variable-length records, record length is invalid if the length of the record is not the same as the record length specified in the block count control field.

When the program is given control, general register 0 contains the address of the block containing the error. The EXRTN macro must be issued if processing is to continue. If register 0 contains X'00000001', the current block will be skipped and processing continues from the next block. Any other code indicates that this block is to be processed as if no error had occurred.

If the WLRERR entry is omitted and the DMS detects an invalid record length, one of the following two conditions occurs:

1. If the FCB entry ERROPT is specified for the file, the record containing the invalid length is treated as an errored block and processed according to the user program's specifications for an error.

2. If the FCB entries ERROPT und ERRADDR are not specified, the job is terminated.

The WLRERR entry does not apply to format-U records. Records of undefined length are not checked for invalid length.

Note:

Registers 14, 15, 0 and 1 are DMS parameter registers. Therefore, unless explicitly described otherwise, it cannot be assumed that these registers have a defined value when the user is given control at the EXLST exit.

The following tables indicate which EXLST exists can be used, and when:

Error exit	STD SAM	NSTD SAM	ISAM	PAM	BTAM	The user can issue		
						CLOSE	EXRTN	Action macro
CLOSER	A	A	A	A	A	N	N	N
DLOCK	N	N	N	A	N	A	N	A
DUPEKY	N	N	A	N	N	A	N	A
EOFADDR	A	A	A	A	A	A	N	A
ERRADDR	A	A	A	A	A	A	N	A
ERROPT	A	A	N	N	N	A	A	N
ISPERR	N	N	A	N	N	A	N	A
LOCK	A	A	A	A	A	N	N	A
NODEV	A	A	A	A	A	N	N	N
NOFIND	N	N	A	N	N	A	N	A
NOSPACE	A	N	A	N	N	A	N	With SAM:N, otherwise A
OPENC	A	N	A	A	N	N	N	A (only VERIF is practical)
OPENER	A	A	A	A	A	N	N	N
OPENX	A	A	A	A	A	A	A	N
OPENZ	A	A	A	A	A	A	A	N
PASSER	A	A	A	A	A	N	N	N
PGLOCK	N	N	A	A	N	A	N	A
SEQCHK	N	N	A	N	N	A	N	A
USERERR	A	A	A	A	A	A	N	A
WLRERR	A	A	N	N	N	A	A	N

Table 5-1: Use of error exits (I)
See Table 5-2 for an explanation of the abbreviations.

Error exit	COMMON exit, if no exit specified	FCB key (field ID1XITB) X' '	1) Termination	Remarks
CLOSER	Z	2C	A	File is considered closed
DLOCK	Z	3C	A	No further locks until all existing locks have been released
DUPEKY	Z	54	A	
EOFADDR	Z	40	A	
ERRADDR	Z	44	A	Error code and/or end byte (PAM and BTAM)
ERROPT	N	48	A	End byte stored in FCB
ISPERR	Z	50	A	
LOCK	Z	10	A	
NODEV	Z	14	A	
NOFIND	Z	58	A	
NOSPACE	Z	4C	A	
OPENC	N	68	N	
OPENER	Z	08	A	Error code stored in FCB
OPENX	N	04	N	
OPENZ	N	18	N	
PASSER	Z	0C	A	
PGLOCK	Z	38	A	
SEQCHK	Z	60	A	
USERERR	Z	5C	A	Error code stored in FCB
WLRERR	N	64	N	Error code stored in FCB

Table 5-2: Use of error exits for disk files (II)

Where:

A = allowed
Z = applicable
N = not applicable
(1) Program terminates if exit was not specified and action was taken

EXRTN RETURN FROM ERROR ROUTINES (TYPE R)

In certain cases, it is necessary to return to the DMS after executing certain EXLST exits in order to continue processing. This is effected by means of the EXRTN macro.

This macro is permissible for the following EXLST exits:

- ERROPT
- OPENX
- OPENZ
- WLRERR

Operation	Operands
EXRTN	$\left\{\begin{matrix}\text{fcbaddr}\\ \\ (1)\end{matrix}\right\},\left\{\begin{matrix}0\\1\\(0)\end{matrix}\right\}$

fcbaddr This operand specifies the address of the FCB of the file whose processing resulted in the error exit (address output area).

(1) Register 1 contains the FCB address.

0 The current block is to be processed as if no error had occurred.

1 The current block is to be skipped and the next block processed.

(0) Register 0 contains the value 0 or 1.

Note:

The second operand is significant only in the case of the ERROPT und WLRERR exits.

FCB CREATE FILE CONTROL BLOCK (TYPE 0)

The FCB macro reserves space for the File Control Block and, optionally, supplies information to it. Additionally, this macro reserves space for the logical routines needed to process the file.
The File Control Block is the central source of information for all access methods (except EAM). It is always the same size, and always has the same field structure.

Format for disk files

Operation	Operands
FCB	[BLKSIZE={STD | (STD,absexp)}] [,DUPEKY=YES] [,EXIT={(relexp) | relexp}]
	[,FCBTYPE={ISAM | SAM | PAM}] [,FILE=filename] [,FORM=SHORT]
	[,IOAREA1={NO | relexp}] [,IOAREA2={NO | relexp}] [,IOREG=regno]
	[,KEYARG=relexp [,KEYLEN=absexp] [,KEYPOS=absexp]]
	[,LINK=linkname] [,LOGLEN=absexp] [,OPEN={INPUT | OUTPUT | EXTEND | UPDATE | INOUT | OUTIN}]
	[,OVERLAP=YES] [,PAD=absexp] [,PAMREQS=absexp]
	[,PAMTOUT=absexp] [,PASS=password] [,RECFORM={{F | V | U} | ({F | V | U} [,{A | M | N}])}]
	[,RECSIZE=absexp] [,RETPD=days] [,SHARUPD={YES | NO | WEAK}]
	[,VALLEN=absexp] [,VALPROP={MAX | MIN}] [,VARBLD=regno]
	[,WROUT={YES | NO}] [,WRCHK={YES | NO}]
	[,OPTION={code | (code,code)}]

BLKSIZE= This operand specifies the size of the file's buffer.

=STD (Form 1):

This operand specifies that the buffer is one PAM block (2048 bytes) in length.

=(STD, absexp) (Form 2):

This entry specifies that a buffer is the number of PAM blocks specified in "absexp" (absolute expression, which may be any value from 1 to 16). The maximum value of "absexp" is 16, which allows a buffer length of 32768 bytes.

Programming notes:

- If this operand is used at OPEN time (for example, to allocate buffers), BLKSIZE=STD (form 1) is always assumed by the PAM action macros.
- No logical record can exceed BLKSIZE. No ISAM variable-length or fixed-length logical record and no SAM variable-length logical record can exceed BLKSIZE-4.
- Strictly speaking, this operand defines the buffer size (i.e. the volume of data transferred to or from an I/O device).

Example:

BLKSIZE=(STD,3)

The buffer is 2048 x 3 = 6144 bytes long; the physical block size is 2048 bytes. It should be noted that a logical record can be up to 6144 bytes long (which implies that logical records can cross physical block boundaries).

There are two principal reasons why the user might specify (STD,absexp) instead of STD.

1. The logical record exceeds 2048 bytes.
2. The logical record is uneconomical for a block size of 2048 bytes. If the user has fixed-length (format-F) records of 1500 bytes and if BLKSIZE=STD is specified, then 548 bytes of each block are wasted.

   However, if the user specifies BLKSIZE=(STD,3) then 6000 of the 6144 bytes (i.e. 2048 x 3) are used, which wastes only 144 bytes per three blocks. The use of very large block sizes, however, increases paging activity.

   When specifying forms 1 or 2 for BLKSIZE, the user does not have to be concerned about the key. The key is always read from or written into the FCB.

   Default value: STD (form 1)

DUPEKY= This operand specifies that duplicate keys are permissible.

If this operand is omitted and duplicate keys are encountered, control is passed to the address supplied in the user's EXLST macro (default value).

EXIT= This operand specifies the address of an exit routine. If the value of this operand is enclosed in parentheses, the address is actually the address of the user program's exit routine. If the value of this operand is not enclosed in parentheses, the address is assumed to be the address of the EXLST macro.

It should be noted here that if the exit is taken, regardless of its type, an indicator is set in a byte in the FCB so that the user routine can identify the specific exit condition.

Moreover, if the exit is taken because of a hardware error during I/O, a 5-byte area in the FCB is also set to contain the following bytes from the CCB:

- standard device byte
- sense bytes 1, 2, 3
- Executive flag byte.

See Appendix A.3 (Table A.3-1) for details of the sense bytes. The Executive flag byte and standard device byte are defined in the CCB DSECT (IDCCB).

Warning:

In the event of a non-specified condition, a branch is made to the location defined by the COMMON operand in the EXLST macro.

If any other conditions are specified by means of appropriate operands, a branch is made to the relevant exit. The COMMON operand saves the programmer the effort of writing a lengthy EXLST macro and thereby generating a large table of error exits.

Example:

```
EXLST OPENX=LOC1,COMMON=LOC
```

The OPENX condition causes control to be passed to LOC1. All other conditions (for example, EOFADDR, DUPEKY) cause control to be passed to LOC.

Default value:

No exit address is created; consequently, any undefined conditions will cause abnormal program termination.

FCBTYPE= This operand specifies the access method to be used to process the file:

ISAM

SAM

PAM

FILE= This operand specifies the fully qualified file name of a file, a generation (with absolute or relative generation number), or the name of a temporary file.

The name of a file generation group is not permitted.

Default value: FCB name (i.e. symbol in the name field).

FORM=SHORT This operand specifies that space is not to be reserved for the corresponding logical routines. Such routines are required by SAM and ISAM for blocking and deblocking records in the I/O buffer.

Default value:

Space is reserved where necessary. It should be noted that PAM never requires the logical routines in the user program; accordingly, this operand is ignored for this access method.

Note:

The SHORT operand is intended for certain applications only. If it was specified for normal ISAM or SAM processing, the FCB cannot be opened.

IOAREA1=NO This operand specifies that no buffer (area) is to be assigned at OPEN time. This value cannot be specified for SAM and ISAM.

=relexp This operand specifies the address of the buffer in order to enable the file to be processed. If this area is less than or equal to 4096 bytes (one page), it must be fully contained on one page and be aligned on a word boundary. If this area is longer than one page, it must:

1. begin on a page boundary
2. be virtually contiguous
3. not cross a segment boundary

Default value:

The DMS dynamically requests buffer space (class 5 memory) at OPEN time.

IOAREA2=NO This operand specifies that no second buffer is to be allocated at OPEN time. The user is cautioned that if this value is specified for SAM, the system cannot perform any overlapping I/O operations. The value cannot be specified for ISAM.

=relexp This operand specifies the address of the buffer, in order to enable the file to be processed. If this area is less than or equal to 4096 bytes (one page), it must be fully contained on one page and be aligned on a word boundary. If this area is larger than one page, it must:

1. begin on a page boundary
2. be virtually contiguous
3. not cross a segment boundary

Default value:

The DMS dynamically requests buffer space (class-5 memory) at OPEN time.

Programming notes for IOAREA:

When a file is opened, the IOAREA addresses are created (if necessary) and verified. They are then moved to system memory. Consequently, any changes concerning the addresses are completely ignored in the FCB.
For new addresses to become effective, it is necessary to close the FCB and to issue another OPEN macro.

This method of processing minimizes internal system processing time (overhead), as the system need not check IOAREA addresses prior to each action macro to ascertain whether they have been modified. PAM und BTAM (see the "DMS Tape Processing" manual) also allow the user to specify buffer areas in their action macros. These buffer addresses are, of course, validated every time the action macros are issued. Additionally, PAM and BTAM allow the user to select IOAREA1 and/or IOAREA2 in their action macros.

During the period when a file is open, the user must not modify the IOAREA except in connection with an action macro. If a file is opened in a class 1 program, IOAREA1 and IOAREA2 must be specified as "relexp" or NO.

If a SAM file is opened in UPDATE mode, the IOAREA2 buffer is not used.

If NO is specified for IOAREA1, the value for IOAREA2 must also be NO. If IOAREA1 is specified as "relexp", the value specified for IOAREA2 must be NO or "relexp". If no value is specified for IOAREA1, the same must apply to IOAREA2, or NO must be specified.

Alignment of the buffer is only necessary if its address is greater than X'800',

IOREG= This operand specifies the general register (2 through 12) which contains the address of the current record (absexp). This implies that records are to be processed in locate mode (applies to SAM and ISAM only).

Default value:

Move mode is assumed.

KEYARG= This operand specifies the address of the key. This applies to certain ISAM action macros (GETKY, ELIM, GETFL, SETL).

KEYLEN= This operand specifies the number of bytes in the record key. The key must be of the same length in every record (1 - 255 bytes)

Default value: 8 bytes

KEYPOS= This operand specifies the position of the first character of the record key within the record. The key must be located in the same position of every record. The first byte in the record must be 1. If format-V records are specified, the 4-byte control prefix is considered to be part of the record.

Default values:

1 for format-F records;

5 for format-V records.

LINK= This operand specifies the file link name (up to eight characters). It provides the link between the FCB and the TFT if data from a FILE command/macro is to be used to modify the FCB. Thus, unique file link names must be provided for the FCBs of a program if modification of these FCBs is planned.

If the file link name for an FCB is omitted, a link name consisting of blanks (X'40') is generated. Specifying "LINK=,..." (null operand) causes the generation of a link name consisting of binary zeros. This prevents the FCB being modified by a FILE command/macro. In this case the OPEN macro creates a TFT entry containing blanks or binary zeroes as the link name.

It is therefore not advisable for the user to omit the link name or to specify the null operand.

LOGLEN= This operand specifies the length of the logical flag for a flagged ISAM file.

Default value: 0

OPEN= This operand specifies in which mode the FCB is to be opened. It should be noted that the OPEN macro can override this specification.

Valid OPEN processing options for the various access methods are shown in the table below (disk files only):

OPEN=	SAM	ISAM	UPAM
INPUT	X	X	X
OUTPUT	X	X	-
EXTEND	X	X	-
UPDATE	X	-	-
INOUT	-	X	X
OUTIN	-	X	X

OVERLAP= This operand specifies that buffered or overlapped read operations are desired when more than one data buffer exists.

This applies to the ISAM macros GET und GETR.

Default value:

No I/O operations are performed in advance, so as to bring about overlapping. If this operand is not specified, ISAM uses IOAREA2, not for anticipatory I/O but as a work area.

PAD= This operand specifies the percentage of buffer length to be reserved during creation of an ISAM file for later insertion or expansion of records in the data blocks.

Default value:

Fifteen percent of the data space is reserved.

PAMREQS= This operand specifies the maximum number of asynchronous I/O requests that the user can issue for this PAM file at any given time. This maximum can be any value from 1 to 100.

Default value: PAMREQS=1

PAMTOUT= This operand specifies how many seconds a job is to wait for one or more PAM locks. Should the requested locks fail to become available after that period, a branch is made to the EXLST exit DLOCK or PGLOCK. The minimum value of this operand is 0; in this case, control is returned immediately, regardless of whether or not the requested locks are available.

The maximum value is 43200 (seconds).

Default value: PAMTOUT=0

PASS= This operand specifies the password needed to access the file (absexp). The password can be up to 4 characters long. This field is filled with zeros to the left or, if it is longer than 4 characters, it is truncated on the left. The method is the same as the one utilized for address constant processing by the Assembler;
e.g. the expansion is DC AL4 (password).

Examples:

PASS=C'ABC'	gives $(00C1C2C3)_{16}$	in the FCB
PASS=X'0102'	gives $(00000102)_{16}$	in the FCB
PASS=C'ABBCDEF'	gives $(C3C4C5C6)_{16}$	in the FCB
Default value:	$(00000000)_{16}$	

RECFORM= This operand specifies the record length type.

=F Fixed-length records.

=V Variable-length records.

=U Records of undefined length. This value is not permissible for ISAM. The combinations (U,A) und (U,M) have no effect.

"A" specifies that the first data byte is an ASA control character;

"M" specifies that the first data byte is interpreted as a control character for EBCDI code.

"N" specifies that no print control character is present.

Although the PAM File Control Block accepts this operand, it is ignored by the action macros.

RECSIZE= This operand specifies the length in bytes of a logical record for format-F records. If RECFORM=V is specified, the RECSIZE entry is not verified; the maximum record length is here always equal to the buffer size.

For format-U records, this operand specifies the general register (2 through 12) which is to contain the length of each record. Whenever a format-U record is read, the DMS supplies its length in this register. If a record of undefined length is to be written, the user must place the length of the record in this register.

Although the PAM File Control Block accepts this operand, it is ignored by the PAM action macro. Nevertheless, it can still be specified, as a file created by PAM can be processed by SAM.

RETPD= This operand specifies the period (in days) during which a file is to be retained, and stipulates when overwriting, updating or erasing of the file is permitted. However, it cannot be used automatically to erase a file from the system (absexp) once the retention period has expired.

"days" must be a decimal integer, the maximum value of which is calculated from the difference in days between the end of the century (31.12.99) and the current date.

Default value: 0 days (i.e. the file can be overwritten or erased immediately).

SHARUPD= This operand specifies whether a UPAM or an ISAM file can be shared by more than one job for the purpose of simultaneous updating.

=YES Shared updating is possible.

=NO Shared updating is not possible.

=WEAK This entry is permitted for UPAM files only.
WEAK ensures write protection. Only one user at a time can write to the file.
Read protection remains the responsibility of the user, who has the following options:

* he can implement read protection by means of his own programming measures.

* he can assume that a record read for the second time will have been modified in the meantime.

The following table illustrates the various competing levels, together with the types of protection offered in each case:

SHARUPD options	Number of users who can			Protection type	
	read	and/or	write	READ	WRITE
YES	N	and	M	*	*
WEAK	N	and	1	-	*
NO	N	or	1	*	*

Notes:

- The operand WEAK is supported for PAM files only.

- With SHARUPD=WEAK, a user job can run either on one processor, or on two different processors that are linked via shareable private disks (SPD).

- The SHARUPD combinations permitted in the case of one or more processors are specified in section 6.2.1.

- If FCBTYPE ≠ PAM, the WEAK operand is processed as NO (i.e. SHARUPD=NO).

VALLEN= This operand specifies the length of the value flag that can be specified for ISAM files in addition to the ISAM key.

If LOGLEN or VALLEN is specified,

KEYLEN + LOGLEN + VALLEN must be less than or equal to 255.

Default value: 0

VALPROP= This operand specifies how the value flags in ISAM files are to be entered in the index blocks:

=MAX The highest value flag from all the records in the data block is entered in the index entry.

=MIN The lowest value flag from all the records in the data block is entered in the index entry.

VARBLD= This operand specifies the general register (2 through 12) in which the DMS stores the amount of free space in the block to be written (absexp). This specification is necessary for SAM files when format-V records are to be created in locate mode.

WROUT=YES The contents of ISAM buffers (data and index blocks) are to be written back immediately after they have been modified.

=NO Writing back is not performed automatically after each modification.

Notes:

- The WROUT function enhances security when processing ISAM files. In the event of a system crash only the records processed by the last macro can be corrupted or lost. (Exception: After a PUT macro, a buffer is written back only when completely filled).
- The increased number of I/O operations reduces throughput.
- The WROUT function takes effect after the ISAM action macros STORE, ELIM, INSRT, PUTX and PUT (after PUT, only when the buffer is completely filled).
- ISAM SHARED UPDATE includes the WROUT function, so in this case the operand is irrelevant.
- A value other than YES or NO will always result in an error message.
- The value in the FILE command or the FILE macro has priority over the value in the FCB macro.
- The operand value specified in the FCB macro applies only if no operand, or a null operand (i.e. WROUT=,...), is entered in the FILE command or the FILE macro.

WRCHK=YES A read-after-write check is performed for all data. This means that written records are checked only for their readability and not for their contents.
As a result, write errors on the disk are recognized in good time and can be recovered by taking the appropriate measures. If the error cannot be eliminated, a branch is made to the ERRADDR routine (via EXLST).

Notes:

- The read-after-write check requires an additional turning of the disk, which adversely affects system performance.
- This specification is not contained in the catalog entry of the file, i.e. it must be entered explicitly each time a file is to be processed and be available when the file is opened.

WRCHK=NO No read-after-write check is performed.

OPTION=code

This operand can be used to specify a list of options. The following can be specified for "code":

GLODEF

If the file name is specified without an explicit user ID and the file cannot be found under the user ID of the person making the call, a second read attempt is performed under the ID which was defined in the class 2 option DEFLUID (see the "System Generation" manual).
Cf. the sections on "Retrieving Files" and "Opening Files".

WARN

In FCB field ID1WCB, return codes are stored as warnings, without execution being interrupted (exit routine).
This means, for example, that screen masks are no longer destroyed and unassignable messages are no longer encountered when procedures are executing.
The user himself decides when and if to evaluate the corresponding field in the FCB and how to respond.
The following warning codes are stored at the present time:

044F Public space limit exceeded, when enforcement is permitted.
0440 Public space saturation level 3 exceeded.

The default value specifies that no codes are stored.

Return information

Return code 0D33 stored in FCB field ID1ECB by the OPEN command has the following meanings, depending on what specifications the user has made in the file name and in the OPTION operand.

File name	OPTION	Meaning of 0D33
:C:$USER.FILE	any specification	The file does not exist under the ID $USER on PVS :C:
$USER.FILE	any specification	The file does not exist under the ID $USER on the default PVS of $USER.
$FILE	any specification	The file does not exist under the ID generated for DEFLUID on the PVC specified therein.
:C:$FILE	any specification	The file does not exist under the ID generated for DEFLUID on the PVS :C:
FILE	not GLODEF	The file does not exist under the caller's ID on the default PVS.
FILE	GLODEF	The file exists neither under the caller's ID on the default PVS nor under the ID generated for DEFLUID on the PVS specified therein.
:C:FILE	not GLODEF	The file does not exist under the caller's ID on PVS :C:
:C:FILE	GLODEF	The file exists neither under the caller's ID nor under the ID generated for DEFLUID on PVS :C:

If the user is given control in an OPEN exit (OPENX, OPENZ, OPENC), the complete file name (with CATID and USERID) is in field ID1FILE of the P1FCB at this time. This is also the case following successful completion of OPEN. Following an OPEN error or following a CLOSE on the FCB, field ID1FILE is again given the file name specified by the user.

Any modification of the file name in the ID1FILE field within an OPEN exit routine is ignored.

This behavior may be modified for users whose programs expect the file name in the FCB to be unchanged.

Programming notes:

When an FCB (file) is opened, most of the processing operands are moved into system memory (P2FCB) after creation and verification. Any change in these operands is therefore ignored. New values become effective only by closing the FCB and reissuing the OPEN macro. Changes to the FCB of an opened file may cause abnormal program termination or unpredictable file processing.

Null operands such as "FCBTYPE=,..." do not initiate the default function. In this case, the Assembler listing indicates that an operand was not fully defined at assembly time and must be specified at execution time.

FCB operands	NULL operand valid	SAM	ISAM	PAM	Operands valid in the FILE command	Operands which may be supplied via the catalog
BLKSIZE	X	X	X	I	X	X
DUPEKY			X	I	X	
EXIT		X	X	X		
FCBTYPE	X	X	X	X	X	X
FILE		X	X	X	X	
FORM		X	X	I		
IOAREA1	X	X	X	X		
IOAREA2	X	X	X	X		
IOREG		X	X			
KEYARG			X	I		
KEYLEN	X		X	I	X	X
KEYPOS	X		X	I	X	X
LINK	X	X	X	X	X	
LOGLEN	X		X	I	X	X
OPEN		X	X	X	X	
OVERLAP			X		X	
PAD			X	I	X	
PAMREQS		I	I	X		
PAMTOUT		I	I	X		
PASS		X	X	X		
RECFORM	X	X	X	I	X	X
RECSIZE	X	X	X	I	X	X
RETPD		X	X	X	X	
SHARUPD	X		X	X	X	
VALLEN	X		X	I	X	X
VALPROP	X		X	I	X	X
VARBLD	X	X	X			
WROUT	X		X		X	
WRCHK		X	X	X	X	
OPTION	X	X	X	X		

Table 5-3: Summary of FCB macro operands valid for each access method (for disk files only)

Where:

I = operand accepted but ignored by action macros
X = operand may be specified

Note:

If an operand is omitted, the appropriate default value is assumed. If a null operand is specified (i.e. the operand value assigned is the blank character string, e.g. in the form FCBTYPE=,) it is assumed that the operand value will be supplied via a FILE command/macro, or from a file catalog entry.

Exception:

Operand LINK=

FCBAD – CREATE FCB ADDRESSES (TYPE 0)

For all DMS macros following each other statically in a program the code is generated in such a way that the File Control Blocks may be symbolically addressed outside the base register. The FCB addresses are here stored in the literal pool. This macro thus facilitates program conversion from BS1000 to BS2000.

Operation	Operands
FCBAD	

OPEN OPEN FILE (TYPE R)

A file must be opened by means of the OPEN macro before it can be processed.

Operation	Operands
OPEN	{fcbaddr / (1)} [,{mode / (0)}]

5

fcbaddr This operand specifies the address of the File Control block.

(1) The address of the FCB is in register 1 (addexp).

mode This operand specifies the OPEN processing mode.

(0) The mode is in coded form in the least significant byte of register 0.

Mode	Code
(not specified)	X'00'
INPUT	X'01'
REVERSE	X'02'
OUTPUT	X'04'
EXTEND	X'08'
UPDATE	X'10'
INOUT	X'20'
OUTIN	X'40'

Default value:

The OPEN mode is specified in the FCB or in the TFT; otherwise INPUT is assumed.

Note:

The characteristics of the various OPEN modes are described under the appropriate access method.

6 ACCESS METHODS

6.1 GENERAL

6.1.1 Overview of the Purpose and Functions of Access Methods

What purpose do access methods serve?

The DMS access methods transfer data between a file and the address space of a job when requested by the job to do so and when the file concerned is open.

Technically speaking, this signifies the transfer of data between the peripheral storage facilities and the main memory of the CPU, with the DMS assuming responsibility for the complicated handling of devices and offering the user interfaces for accessing:

- logical records:
  These are the elements of the logical structure known as a file;

- buffer contents:
  From the program's point of view, these are the units of data transfer and storage (see also sections 6.1.2, "Relationships between Access Methods" and 6.1.4, "Relationship between Logical Record, Buffer and PAM Block").

How do access methods function in principle?

When a user program accesses the contents of a file, the following processes can be distinguished:

- The DMS transfers data blocks between peripheral storage and main memory. This operation is essential for all access methods.

- That part of main memory which accepts one or more data blocks or from which data blocks are transferred to peripheral storage is called a buffer. The buffer belongs to the address space of the job which initiated the I/O operation.

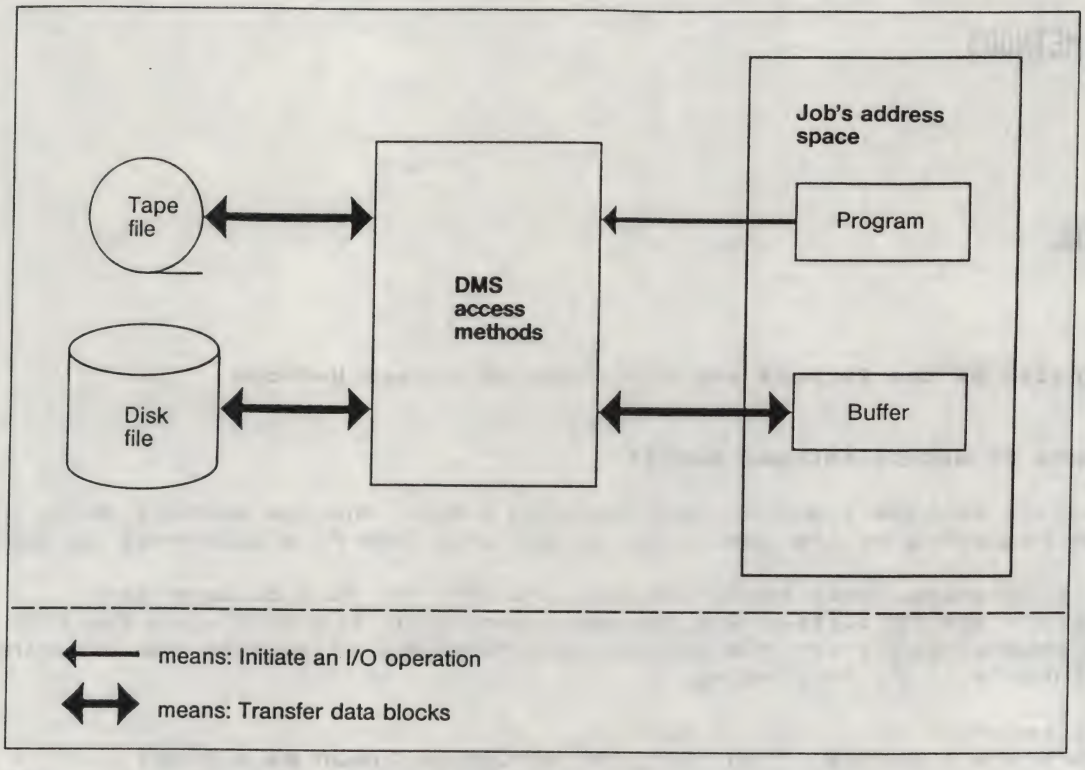


Fig. 6-1: Transferring data blocks

- The following procedure depends on whether access is block-oriented or record-oriented:

  In the case of block-oriented access, the user program and the DMS need only know which is the current buffer if alternate buffer operation was specified when the file was opened. (Alternate buffer operation serves to accelerate processing; the program processes the contents of one buffer while the DMS makes available or writes out another buffer).

In the case of record-oriented access, the following facilities are provided:

- **Move mode**

  The DMS moves a logical record between the buffer and a work area belonging to the program. The program can freely select this work area whenever it accesses a logical record.

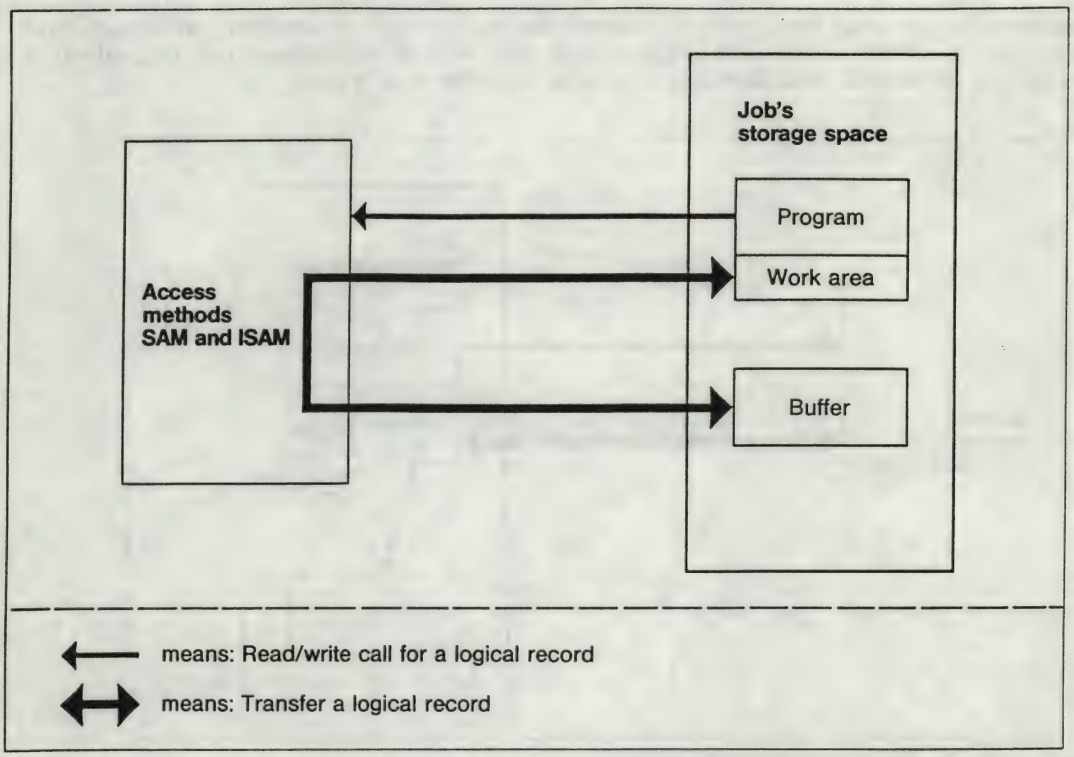


Fig. 6-2: Move mode for record-oriented access methods

- **Locate mode**

  The DMS transfers to the program an address which points to a position in the buffer, either from which the program can read a record it has located, or to which it can write a record. A register for transferring the address must be specified when the file is opened (see the operand IOREG=regno in the FCB macro).

  Note:

  Access to a logical record requires a buffer transfer only if the current buffer does not contain the record required or if, when a file is created sequentially, the buffer is full.

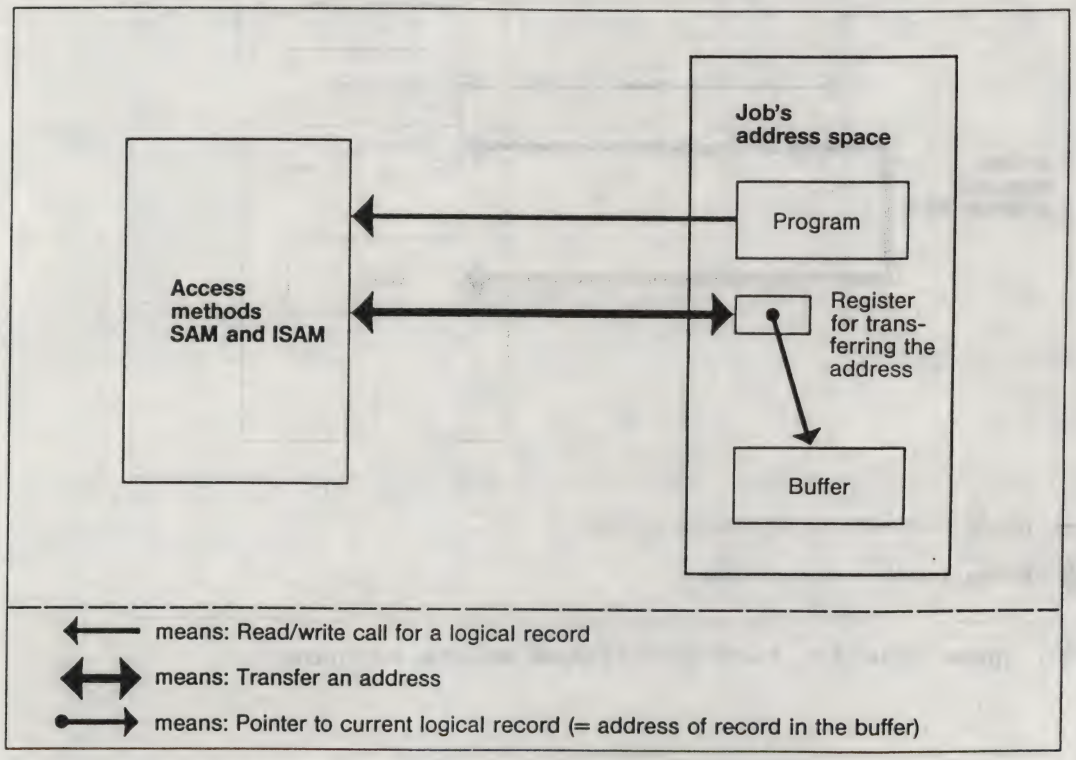


Fig. 6-3: Locate mode for record-oriented access methods

Selecting an access method

The selection of an access method depends on the particular requirements of the problem to be solved.

The DMS offers the user the following access methods:

Block-oriented	Record-oriented
UPAM	ISAM
EAM	SAM

The following table is intended as an orientation aid in selecting an access method.

Access methods	Requirements							
	Access				Record type			
	direct	sequen-tial	record-oriented	block-oriented	V	F	U	
UPAM	X			X		X		
		X		X		X		
ISAM	X		X		X			**
	X		X			X		
		X	X		X			
		X	X			X		
SAM	X	X	X		X			
	X	X	X			X		
	X	X	X				X	
EAM *)		X		X		X		
	X			X		X		

Fig. 6-4: Suitable access methods for specific requirements

*) Suitable for work files only (see section 6.5, "EAM").

Example of how to use the table:

To find a BS2000 access method which satisfies the following requirements:

- Disk file processing
- Record-oriented access
- Record type = variable length

The line followed by "**" satisfies these requirements; i.e. ISAM.

6.1.2 Relationships between Access Methods

The central DMS access method is privileged PAM (Privileged Primary Access Method = PPAM).

Privileged PAM operates with a buffer length of 2048 bytes or an integral multiple of 2048 (maximum length = 16 x 2048). It can process only those disks that are preformatted into blocks of 2048 bytes in length (see also the "System Controller's Guide" and "System Generation" manuals). These blocks are also called PAM blocks.

Privileged PAM operates in association with the following access methods:

- UPAM
- EAM
- ISAM
- SAM (when accessing disk files and tape files with standard blocks)

Access method BTAM, on the other hand, operates independently of privileged PAM; this also applies to the access method SAM when accessing tape files whose block lengths are incompatible with the block length of privileged PAM. This relationship is shown in the following diagram.

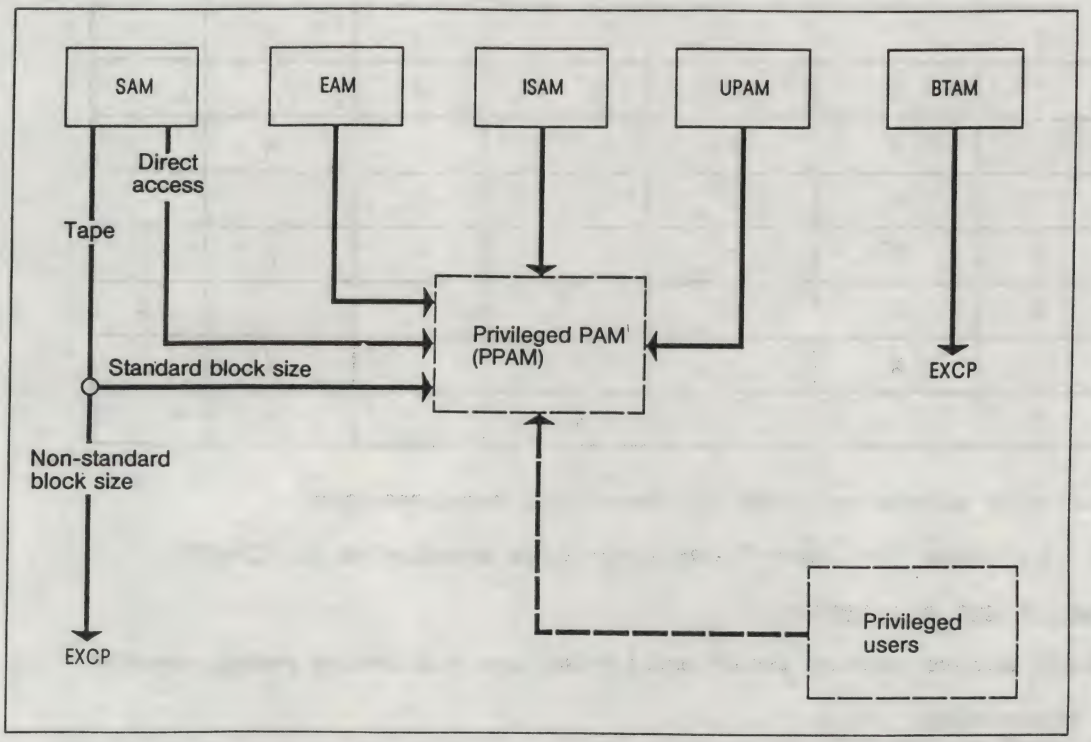


Fig. 6-5: Relationships of access methods to privileged PAM (PPAM)

6.1.3 Logical Record Formats

Three types of logical record format may be distinguished:

Format F (fixed-length records)

Records having this format appear as follows:

```
<------------------------ User data ------------------------------>

|                                                                  |
Byte 0                                                   Last byte
```

A file is designated as having "fixed-length records" if it was set up with the specification RECFORM=F. The record length should be defined with the operand RECSIZE=length (see the FILE command and FCB macro).

Format V (variable-length records)

Records having this format appear as follows:

```
| RL | Reserved | <------------- User data ------------------------>

|    |          |                                                  |
Byte 0 Byte 2   Byte 4                                   Last byte
```

RL: Record length = Length of user data + 4

A file is designated as having "variable-length records" if it was set up with the specification RECFORM=V (see the FILE command and FCB macro).

Format U (records of undefined Length)

Records having this format appear as follows:

```
| <------------------------------- Buffer -------------------------------> |
|-----------------------------------------------------------------------|
| <------------- User data -------------------> | Undefined              |
|-----------------------------------------------------------------------|

|                                              |
 Byte 0 of buffer                              Last byte
       =                                       (see text)
 Byte 0 of record
```

To process records of undefined length, the user must specify a register when the file is opened which will contain the currently applicable length at any particular time (see the operand RECSIZE=regno in the FCB macro). A file is designated as having records of undefined length if it was set up with the specification RECFORM=U (see the FILE command and FCB macro).

6.1.4 Relationship between Logical Record, Buffer and PAM Block

As far as the DMS is concerned, a logical record is a data element that cannot be further subdivided. This applies both to searching in a file -- in the case of record-oriented access -- and to transferring data to and from the calling program. This means that a buffer must be at least as long as the longest record in a file, a point that should be observed when selecting the buffer length.

The buffer length must always be compatible with the block format of privileged PAM.

Therefore only the following values are permissible buffer lengths:

n*2048 bytes (where n is an integer in the range from $1 \leq n \leq 16$).

The buffer length can be defined for each file with the operand BLKSIZE=(STD,n); see the FILE command and FCB macro.

Examples of record/buffer/block relationships in SAM and ISAM:

1. Given:

   Format-F records of 100 bytes
   BLKSIZE=(STD,2)
   Buffer size = 2 blocks (4096 bytes)

   The buffer consists of 40 logical records; the rightmost 96 bytes of the buffer are unused. Nevertheless, two 2048-byte blocks are written. It should be noted that record 21 of the buffer is spread over two blocks; the first 48 bytes are in block 1, the last 52 bytes are in block 2.

2. Given:

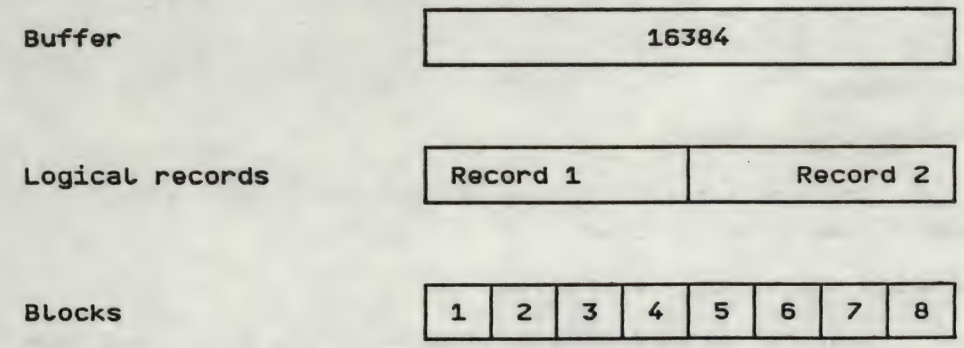
   Format-F records of 1500 bytes
   Standard blocks
   Buffer size = 1 block (2048 bytes)

   This is a bad choice since 548 bytes are wasted. A desirable buffer size would be three blocks (6144 bytes). The buffer could then contain four records, and 6000 out of 6144 bytes could be used. Three blocks, each of 2048 bytes, are written.

3. Given:

   Format-F records of 8192 bytes
   BLKSIZE=(STD,8)
   Buffer size = 8 blocks (16384 bytes)

   Each buffer contains two logical records; each logical record comprises four blocks.



6.2 UPAM (USER PRIMARY ACCESS METHOD)

UPAM ist the primary, block-oriented access method in BS2000 for random (direct) access.
It provides for read or write access (including deletion) to any block in the file at any time.
The block length is 2048 bytes. Record structures within the blocks remain unknown to UPAM.

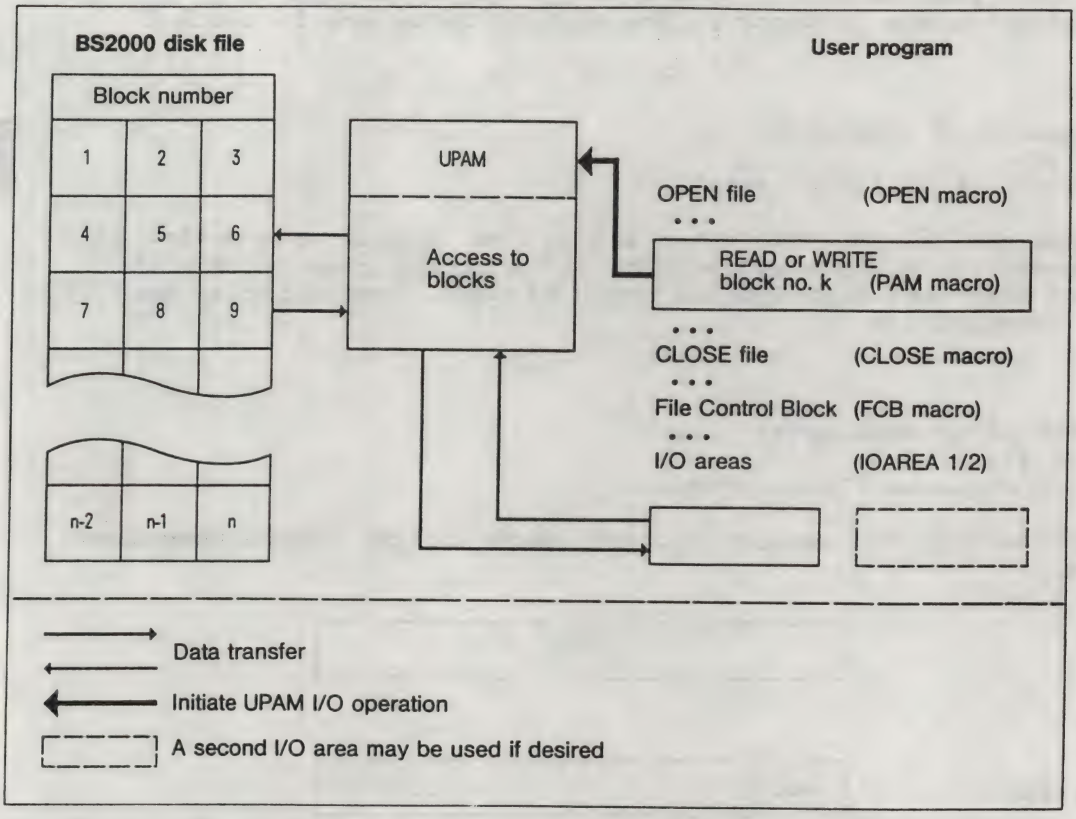


Fig. 6-6: Basic mode of operation of UPAM

UPAM offers the following processing facilities:

- Generation of disk files, not extending over more than one disk. The user himself must program access to logical records in these files (e.g. sequential access or associative access by means of the hash method).

- Reading of SAM and ISAM files (OPEN=INPUT) and their transfer to other volumes (e.g. from disk to tape).
  The file attributes are always stored in the FCB, e.g. BLKSIZE, RECSIZE, RECFORM.
  This enables the user to program access to logical records.

Exception:

A file generated by SAM or ISAM cannot be opened in UPDATE mode for UPAM.
In addition, an ISAM file opened for ISAM in SHARED UPDATE mode cannot be opened for UPAM.

Note:

With ISAM files, the relationships between the indices and the logical records are very complex; accessing logical records requires a correspondingly high degree of programming outlay. As a result, it is impractical to process ISAM files with UPAM.
However, UPAM can be used to transfer an ISAM file, block by block, to tape.

- Shared (file) update

  Multiple parallel jobs can process a UPAM file simultaneously (the file must not have been generated by SAM or ISAM).

- Chained I/O

  A single PAM macro can be used to input/output up to 16 consecutive PAM blocks of a file.

- Chaining of PAM macros in list form

  Up to 255 PAM macros in list form (these need not necessarily all refer to the same file) can be handled by a single UPAM I/O request, i.e. only one SVC is required.
  The chaining of PAM macros serves to optimize the runtime behavior of user programs (as does chained I/O).

- Informing the user job upon termination of a UPAM I/O operation and starting a contingency task (eventing mechanism).

6.2.1 Opening a UPAM File

The file can be opened in the following modes:

1. INPUT: Retrieve blocks from an existing file.

2. OUTIN: Create a new file and, if required, retrieve blocks from this file.
   As a new file is being created, labels are generated.

3. INOUT: Retrieve blocks from an existing file and, if necessary, insert and/or replace records. No labels are generated since the file is assumed to exist already.

UPAM macro functions / OPEN=mode	INPUT	OUTIN	INOUT
RD, RDWT, LRD, LRDWT	X	X	X
WRT, WRTWT, WRTWU	-	X	X
WT, CHK	X	X	X
LOCK, UNLOCK, SETL	X	X	X

Table 6-1: UPAM macro options and permissible OPEN modes

The first user can select any combination of values for OPEN and SHARUPD when opening his UPAM file. Table 6-2a below defines the combinations of OPEN and SHARUPD which another user can use to open a file that has already been opened. If the file was opened by more than one user, then the OPEN/SHARUPD combination of another user is compared with each of the previous ones. For the file to be opened again, each of these comparisons must be successful.

Mono-processor			USER B								
			SHARUPD=YES			SHARUPD=NO			SHARUPD=WEAK		
OPEN mode			INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN
USER A	S.U.=YES	INPUT	X	X		X			X		
		INOUT	X	X					X		
		OUTIN	X	X					X		
	S.U.=NO	INPUT	X			X			X		
		INOUT							X		
		OUTIN							X		
	S.U.=WEAK	INPUT	X			X			X		
		INOUT							X		
		OUTIN							X		

Table 6-2a: Permissible SHARUPD/OPEN combinations

Illegal combinations result in an OPEN error.

Multi-processor			USER B/SYSTEM 2								
			SHARUPD=YES			SHARUPD=NO			SHARUPD=WEAK		
OPEN mode			INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN	INPUT	INOUT	OUTIN
USER A SYSTEM 1	S.U.=YES	INPUT	X			X			X		
		INOUT							X		
		OUTIN							X		
	S.U.=NO	INPUT	X			X			X		
		INOUT							X		
		OUTIN							X		
	S.U.=WEAK	INPUT	X			X			X		
		INOUT							X		
		OUTIN							X		

Table 6-2b: Permissible SHARUPD/OPEN combinations

The term "multiprocessor environment" refers to a network comprising several systems interlinked by means of shareable private disks.

The combinations for accessing from 2 systems are illustrated in the table.

Illegal combinations result in an OPEN error.

6.2.2 Shared File Update

If a UPAM file is to be processed by multiple parallel jobs, the operand SHARUPD=YES must be specified in the FCB in each case (see also section 6.2.1, "Opening a UPAM File").

The following points should be noted with regard to shared update of UPAM files:

- In UPAM, there is no implicit locking/unlocking of PAM blocks. The user must explicitly request or release every lock himself. CLOSE is the only exception: before the file is closed, any PAM blocks still locked for this job in the file are unlocked.

- Locking a PAM block by no means prevents other jobs from reading it or writing to it. What it does prevent is locking of this PAM block by another job. Therefore, when updating the same UPAM file, all jobs should proceed in the following sequence:

  LOCK --> READ --> WRITE --> UNLOCK

  In this way, the integrity of the file is ensured.

  The user should not attempt to request other resources for exclusive use while he has PAM blocks locked. This could lead to a deadlock situation (EXLST output DLOCK).

- No more than 255 PAM blocks can be locked in any single job.

- Within a given job, the same PAM block must not be locked more than once.

  UPAM examines the FCB field ID1TOUT (value of the PAMTOUT operand) whenever one or more locks are requested.

  The length of this interval can therefore be modified dynamically by the user program as long as the file is open. When UPAM receives two or more requests for a lock within an operand list chain (operand CHAIN= in the PAM macro), the PAMTOUT value of the FCB indicated by the first request for a lock in that chain is applicable to this chain of requests. Should no locks be available immediately, the job is queued for the number of seconds specified in the ID1TOUT field. If the locks requested are still not available, either the DLOCK or the PGLOCK exit is taken, depending on whether or not the job has PAM blocks locked at that time.

- Any LOCK or UNLOCK operation applied to a file previously opened with SHARUPD=NO is treated as a NOP.

  <u>Exception</u>:

  File pointer updating.

- If PGLOCK occurs, the program can continue normally, e.g. make a new attempt to lock the block after a waiting time. If DLOCK occurs, a multiple lock has been attempted, if not implemented, and the program becomes unstable. As long as it is in this unstable state, any attempt to lock a PAM block before all current locks are unlocked will cause the program to abort. Only when all current locks have been unlocked, either during the routine for DLOCK or thereafter, will the program become stable again and be able to request locks as usual.

6.2.3 Chained I/O

Chained I/O permits the transfer of up to 16 logically consecutive PAM blocks with a single PAM macro (not to be confused with the chaining of PAM macros in list form with the operand CHAIN=).
Chained I/O reduces the number of I/O operations (and interrupts) and thus saves processing time. However, it does somewhat increase main memory requirements and thus also the amount of paging activity.

Chained I/O is used if the LEN operand is given a value in the range from 2049 through 32768, or if BLKSIZE=(STD,n) is specified.

The following special features should be noted with regard to chained I/O:

- Procedure upon reaching end of file

  If end of file is reached during a read operation, UPAM transfers to the buffer only those PAM blocks belonging to the file.

  UPAM informs the calling job about the end-of-file condition as follows:

  - If eventing is not used:

    The user job receives control at the EXLST exit USERERR with error code X'0922' in the ID1ECB field of the FCB.
    The number of PAM blocks transferred is contained in the ID1NBPP field of the FCB. If the value of this field is X'00', all the PAM blocks to be read are located outside the file. If the value of this field is greater than X'00', the user job must execute a wait operation if this was not implicitly contained in the read operation (i.e. in a RDWT operation).

  - If eventing is used:

    If all the PAM blocks to be read lie outside the file, UPAM passes control to the user job at the EXLST exit USERERR with error code X'0922' in the ID1ECB field of the FCB.
    If at least one of the PAM blocks to be read belongs to the file, UPAM causes either the continuation of the basic task or the initiation of a contingency task (see section 6.2.5). Now the IDECBNPA field of the FECB contains an indication as to the number of PAM blocks that have been transferred.

Contents of IDECBNPA	Meaning
= X'00'	All PAM blocks to be read were transferred to the buffer.
= X'0n'	Only some of the PAM blocks to be read belong to the file and were transferred to the buffer. The value of IDECBNPA is the same as the number of these PAM blocks.

In the case of a WRT operation, a secondary allocation is performed; the specified PAM blocks are appended to the file.

- Locking and unlocking of PAM blocks

  Only the first of a series of PAM blocks to be locked/unlocked need be specified in a PAM macro; the number of blocks to be locked/unlocked is given by the operand LEN.
  It should, however, be noted that after a lock or unlock operation, the file pointer points to the last PAM block that was locked/unlocked. This block may lie outside the file (see also above, "Procedure upon reaching end of file").

- Processing of PAM keys

  There are two possible ways of processing PAM keys:

  - The user reads/writes each individual key of a series of PAM blocks (operand MKEY=YES; KEYFLD must contain the address of a sufficiently large area).

  - The user reads/writes only the first key of a series of PAM blocks.

    On writing, the following blocks are assigned the same key as the first block (the logical block number is simply incremented by 1).

6.2.4 Chaining of PAM macros in List Form

The chaining of PAM macros is implemented by means of the following procedure:

- The macros to be chained are to be generated in list form by means of the operand MF=L and accommodated in a constant area; chaining is effected by specifying the operand CHAIN=.

  Thus, the macros (except the last) have the form:

element	PAM	fcbaddr,operation,...,MF=L,CHAIN=next-element

  In the case of the last element in a chain, the operand CHAIN= is omitted. A chain of PAM macros in list form is called with a PAM macro having the following format:

PAM	MF=(E,first-element)

  Only one SVC instruction is executed for each chain; i.e. the chaining of UPAM requests avoids the overhead of a multiple SVC. The elements of a chain need not necessarily relate to the same file.

Example of coding:

```
          START
          BALR 4,0
          .
          .
          .
          PAM MF=(E,CHN1EL1)
          .
          .
          .
          TERM

*KONSTANTENBEREICH (= constant area)
CHN1EL1 PAM     ......,MF=L,CHAIN=CHN1EL2
CHN1EL2 PAM     ......,MF=L,CHAIN=CHN1EL3
CHN1EL3 PAM     ......,MF=L
          .
          .
          .
          END
```

The following points should be noted with regard to the chaining of PAM macros in list form:

- If execution is successful, the user gains control at the statement following the PAM macro which requested processing of an operand list chain.

- All operations are carried out in exactly the same order in which the PAM macro lists appear within the chain - with one exception:
  When the first operation which requests locking is encountered, the rest of the chain is examined, and all operations which request locking are registered. If it is impossible to implement all the requested locks within the available time, the chain is terminated at the operation which requested the first lock.

- If any process which was requested by an element in an operand list chain was unsuccessful, this process and all remaining processes of the chain (including locks) remain unexecuted; control is passed to the appropriate EXLST exit.
  At this point, register 1 points to the FCB of the file in which the error occurred. The error code (ID1ECB) and the error byte (ID1XITB) are set in this FCB as usual, and the ID1CHERR field in the FCB is set to the address of the element of the operand list chain that caused the error.

- It should be noted that a CHK operation for an unfinished I/O operation also causes control to be transferred from the operand list chain to the user program at the address specified. The remaining operations in that chain (including locks) remain unexecuted, and the ID1CHERR field in the FCB is set to the address of the operand list element in which the CHK took place. It is therefore inadvisable to use check operations on operand list chains.

- It is the responsibility of the user to ensure that, within a chain, appropriate use is made of the lock, read, write, wait, check and release operations. Buffers and key fields are utilized by UPAM according to the request. A check is made to verify the existence of a buffer and for authorized access, but there is no guarantee that a buffer or a key field filled by an operation in a chain will not be overwritten by a later operation of that chain.

- A dummy program section (DSECT), which can be generated with the IDPPL macro, defines the format of the PAM operand list.

- In all cases where a chain of UPAM operand lists cannot be fully processed (e.g. an operation failed, an error was detected, a lock could not be carried out, EOF was encountered, or CHK was issued for a current I/O request), the address of the first unexecuted chain entry is moved to the ID1CHERR field of the FCB. It may be assumed that all preceding entries were executed correctly.

- There are two cases where UPAM cannot inform the user of errors via the FCB/EXLST facility:

  - The UPAM SVC is carried out and neither register 1 nor any chain operand contains a valid address (e.g. the address is not aligned on a word boundary, or the address describes a field which is not entirely part of the user's virtual storage space and is not large enough to contain a UPAM macro operand list etc.).

  - The FCB address in the UPAM macro operand list is either missing or invalid.

  In these two cases, there is no FCB which could be informed of the error; UPAM therefore aborts the program.

  A complete UPAM operand list chain is declared valid before any process is initiated. Should either a CHAIN address or an FCB address be invalid, the job is aborted before any chain element is executed.

  If the eventing mechanism is used and there is no event item available at the end of the I/O operation to which the event can be reported, the user program is likewise aborted.

6.2.5 Informing the User Job upon Termination of a UPAM I/O Operation (Eventing Mechanism)

Eventing with the specified macros is described below; for a more detailed description see the "Executive Macros" manual.

Eventing is used by UPAM to report termination of a requested I/O to a job.

The job can

- be continued in parallel to the UPAM I/O operation and, when the anticipated event occurs (in this case, termination of the requested I/O operation), proceed with a contingency task (asynchronous case).
- wait for termination of the requested I/O operation and then proceed (this type of processing is of course also possible without eventing).

After completion of an I/O operation, UPAM issues a message to the associated event item (with the macro POSSIG).
Sooner or later this message will encounter the request issued by the user (SOLSIG).
When both request and message are present (for the same event item), a contingency task is started or the basic task is continued, as the case may be.

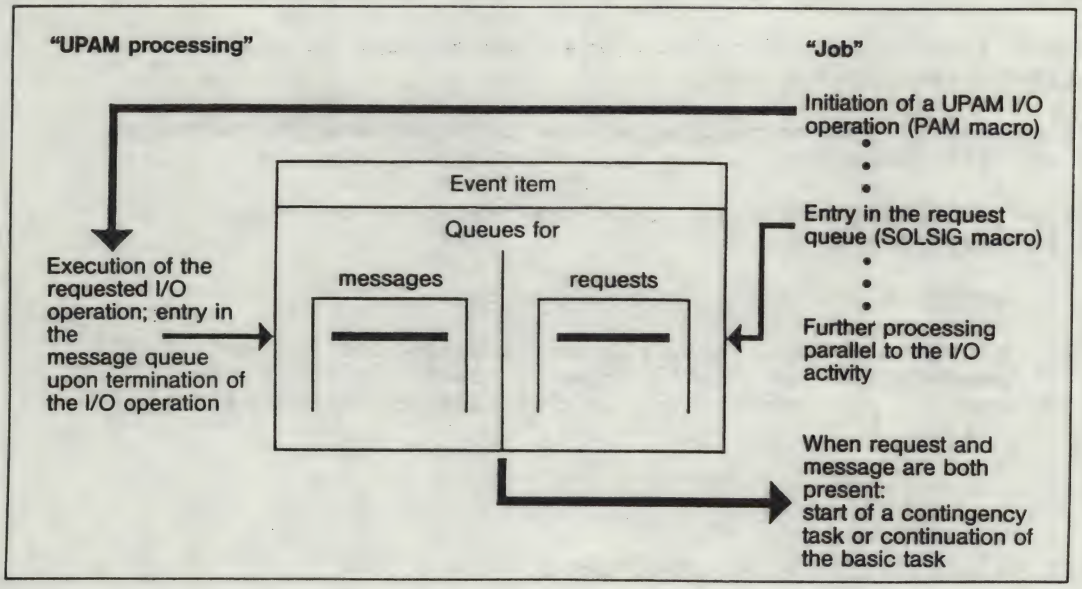


Fig. 6-7: Coordination of user job and UPAM processing

UPAM

Whenever this mode of processing is to be used, the following must occur in the basic task:

- The system must be informed of the event item(s) and contingency definition(s) to be used (macros ENAEI, ENACO).

  The number of event items is equal to the maximum number of UPAM I/O requests simultaneously issued to the system (PAM macros with read or write function: = total of values from the FCB operand PAMREQS=).

  The number of contingency definitions depends on whether different procedures are required after execution of I/O operations. If, for instance, the same procedure is always to apply, it is sufficient to code the contingency definition once only.

- A 14-byte control block (File Event Control Block, FECB) must be set up for each event item.

- The correct value for the operand PAMREQS= must be specified in the FCB macro for each file; this corresponds to the maximum number of I/O operations requested simultaneously for that file.

- The address of the associated control block must be specified for each UPAM I/O request (operand FECB= in the PAM macro). No wait operations must be requested by the PAM macro, either explicitly or implicitly (see also the next point).
  The sequence of instructions READ ——> WRITE ——> WAIT for the same block thus produces an undefined result. The user must wait for the event (termination of the I/O operation) before issuing the WRITE call.

- After each UPAM I/O request, precisely one request must be issued to the associated event item (SOLSIG macro).

  This request serves to determine whether the basic task is to continue in parallel to the I/O operation, or to wait for it to terminate.

  At the start of a contingency task, this task receives the following information in registers 2 and 3:

Register 2	contains the event information code
Register 3	contains in its three rightmost bytes the address of the operand list of the terminated operation (only with chained PAM macros) and in its leftmost byte the value X'10'.

Notes:

- The FECB must be aligned on a word boundary.

- For a detailed description of eventing, refer to the "Executive Macros" manual.

- Until the first I/O operation with an FECB is terminated, this FECB must not be used for other I/O operations.

Format of the File Event Control Block (FECB)

Meaning of field	Field length	Field name
Internal code for event item	4	CBEVID
Address of FCB	4	CBP1LNK
Standard device byte	1	CBSDB
Sense bytes	3	CBSDB1...3
Executive flag byte	1	CBEFB
Number of PAM blocks transferred	1	CBNPA

Table 6-3: Format of the File Event Control Block (FECB)

The FECB can be provided with symbolic names by means of the IDECB macro.

Structure of a contingency definition

In a contingency task the user can respond to the various possibilities of I/O termination.

A UPAM I/O operation can be terminated in a number of different ways:

- Normal I/O termination EFB=X'80'
- I/O operation led to special condition EFB=X'C0'
- Unrecoverable error (e.g. hardware error) EFB=X'A0'

(EFB = Executive flag byte; see FECB)

Multiple branches must be provided in contingency definitions to accommodate the different cases above.

Example:

The example illustrates the possibilities offered by eventing with UPAM.

The program gives rise to the following actions:

1. Open input file and prepare for eventing.
2. Start UPAM input operation.
3. Make entry in event item's request queue.
4. Execute program loop without end condition; count executed loops.
5. Interrupt loop and start contingency task when input operation has terminated.
6. Load program counter of interrupted basic task with a continuation address outside the loop.
7. Terminate contingency task and continue basic task.
8. Output number of executed loops.
9. Terminate eventing and close file.

The above numbering is used in the following source program, with the exception of action 5, which is performed by the operating system.

The program performs actions 2 through 8 ten times in an outer loop (loop counter is register 5).

```
UPAMPRO  START
         PRINT NOGEN
         BALR  4,0
         USING *,4
         LH    5,=H'10'
*
         OPEN  UPAMDAT,INPUT                                  }
         ENAEI EINAME=EINGABEOPENDE,EIIDRET=ESTBLOCK          }
         ENACO CONAME=LADENBEFEHLSZAEHELRBASPROZ,          *  } ①
               COIDRET=COKUKEN,COADAD=ADRCONTI                }
*
STARTEA  PAM   UPAMDAT,RD,FECB=ESTBLOCK,HP=1                  }
ANFORD   SOLSIG EIID=ESTBLOCK,COID=COKUKEN                    } ② ③
         SR    6,6                                            }
*
SCHLEIFE AH    6,=H'2'                                        }
         SH    6,=H'1'                                        } ④
         SH    6,=H'1'                                        }
         AH    6,=H'1'                                        }
         B     SCHLEIFE                                       }
*
NACHCON  CVD   6,ANZAHL                                       }
         UNPK  ANZAHL1,ANZAHL                                 }
         MVZ   ANZAHL1+15(1),ZONE                             } ⑧
         WROUT MELD,TERM,MODE=LINE                            }
         BCT   5,STARTEA                                      }
*
         DISEI EINAME=EINGABEOPENDE                           }
         CLOSE UPAMDAT                                        } ⑨
TERM     TERM                                                 }
*
*
CONTIN   BALR  4,0                                            }
         USING *,4                                            }
         L     6,ADNACHCO                                  *  } ⑥ ⑦
         CONTXT PROCESS=MAIN,FUNCT=WRITE,                     }
               STACKR=(PC),OWNR=(6)                           }
         RETCO
*
*     KONSTANTEN UND STEUERBLOECKE
*
ANZAHL   DS    D
MELD     DC    AL2(MELDEND-MELD)
         DC    X'404001'
ANZAHL1  DS    CL16
         DC    'SCHLEIFENDURCHLAEUFE'
MELDEND  EQU   *
*
COKUKEN  DS    F
ADRCONTI DC    A(CONTIN)
ADNACHCO DC    A(NACHCON)
ESTBLOCK DS    4F
UPAMDAT  FCB   FCBTYPE=PAM,LINK=EIN,                        *
               IOAREA1=EINBER,IOAREA2=NO
ZONE     DC    X'F0'
         LTORG
*
*
*
*    EINGABEBEREICH
*
         ORG   UPAMPRO+2048
EINBER   DS    CL2048
         END
```

```
FILE UPAMPRO.EIN,LINK=EIN,FCBTYPE=PAM
EXEC *
%  P001 -DLL V-725
%  P500 UPAMPRO/001/82-02-05 LOADED
0000000000019268SCHLEIFENDURCHLAEUFE
0000000000026457SCHLEIFENDURCHLAEUFE
0000000000024323SCHLEIFENDURCHLAEUFE
0000000000018992SCHLEIFENDURCHLAEUFE
0000000000025732SCHLEIFENDURCHLAEUFE
0000000000025857SCHLEIFENDURCHLAEUFE
0000000000020617SCHLEIFENDURCHLAEUFE
0000000000026101SCHLEIFENDURCHLAEUFE
0000000000029053SCHLEIFENDURCHLAEUFE
0000000000013818SCHLEIFENDURCHLAEUFF
```

(ANZAHL = number; KONSTANTEN UND STEUERBLOECKE = constants and control blocks; SCHLEIFENDURCHLAUEFE = executed loops)

6.2.6 General Programming Notes

- Since a block is not transferred to the user buffer until an explicit action macro has been issued, a delay is caused.
  Consequently, an asynchronous I/O operation must be terminated with the WT action macro. In the case of P1 eventing, the SOLSIG macro should be used (synchronously or asynchronously).

- For every unsuccessful branch to the UPAM routines, control is transferred to the routine specified in the EXIT operand of the FCB, or to the corresponding EXLST routine. An identifier is stored in the FCB.

- Whenever UPAM causes program termination, it places the address where the termination occurred in register 0; the address of the UPAM operand list chain element in which the error was detected is placed in register 1; and the UPAM error code is placed in register 15. Thus it is easy to locate this information in the printout of the memory dump.

  If register 1 contained an invalid address at the time when the PAM macro was executed for the first time, register 0 of the memory dump will contain that invalid address, and register 1 in the memory dump will contain zeros (i.e. the error occurred before the first element of the operand list chain was located).

- The five EXLST addresses used by UPAM are:

ERRADDR	Hardware error or abnormal I/O termination
USERERR	Invalid use of macro in the program or attempt to read a PAM block not belonging to the file (EOF)
EOFADDR	Attempt to read a *DUMMY file
PGLOCK	Not all of the requested locks are available within the given period of time, and the job has not currently locked any blocks.
DLOCK	Attempt to read a locked block, or request for an additional lock.

- PAM blocks which have been allocated to a file, but which have not yet been written by the owner of this file are identified by their internal file names (bytes 1 through 4 in the PAM key), which do not correspond to the current internal file name.
  Comparison must be performed by the user, who must bear in mind that:

  - At OPEN time, the current internal file name is written to the first word in the ID1KEY1 field of the FCB.

  - After execution of a RDWT operation, the internal file name of the block read is located in the first word of the FCB field ID1KEY2.

  - After execution of one of the operations LRD, LRDWT, RD, WRT, WRTWT or WRTWU, the internal file name of the PAM block concerned is located in the first word of the FCB field ID1KEY1. The entry generated by OPEN is overwritten, but should be saved prior to processing for subsequent comparison.

  - See also the KEYFLD operand in the PAM macro.

- The ID1LWB field in the FCB is used to store the address of the last buffer on which a WT operation was carried out successfully by UPAM. The highest-order byte is used as an indicator. If the last UPAM operation directed to the file was a successful WT (either explicit or implicit; in the latter case, irrespective of the success or otherwise of the operation which initiated that WT), then the highest-order byte of ID1LWB assumes the value X'00', and the three low-order bytes contain the address of the buffer to which the WT referred.

  On the other hand, if the last UPAM operation on that file did not cause a WT, the highest-order byte of ID1LWB is set to X'FF', and the remaining three bytes are without significance.

  The request to lock the block had to be rejected and the user program itself has already locked the block.

6.2.7 UPAM Record Format

UPAM operation is block-oriented. The physical data blocks always have a length of n*2048 bytes (where n = 1,...,16).

If the operand LEN= in the PAM macro has a value not equal to n*2048 (where n = 1,...,16), then:

- in the case of a write operation, the last PAM block to be written is filled with binary zeros.

- in the case of a read operation, the last PAM block to be read is not transferred to the buffer.

Example:

Let the operands WRT and LEN=4000 be specified in the PAM macro.
4000 bytes are transferred from the buffer, 96 bytes containing X'00' are appended, and 2 PAM blocks are written to the file.

1st PAM block	2nd PAM block	
2048 bytes from buffer	1952 bytes from buffer	96 bytes X'00'

6.2.8 PAM Keys

Each PAM block is allocated a 16-byte key (PAM key).

The PAM key has the following structure:

- Bytes 0 through 3 contain the internal file name (coded file ID) of the file to which the PAM block belongs or last belonged.
  The DMS establishes the internal file name on the basis of a number of factors including the file creation date. This provides for unambiguous differentiation between different files.

- Byte 4 contains the version number which the file had when the block was last processed.
  This byte is evaluated and/or updated by the DMS or ARCHIVE when the file is created, updated, deleted or formatted.

- Bytes 5 through 7 contain the logical block number of the PAM block.

- Bytes 8 through 15 contain different information depending on the access method used. UPAM does not utilize this field; nevertheless the user should not use these bytes.
  ISAM and SAM need this field. Further details may be found in sections 6.3.8, "Use of PAM Keys in ISAM", and 6.4.4, "Use of PAM Keys in SAM".

Note:

Certain disk types are initialized with 512-byte blocks without a separate field for the PAM key. For this reason UPAM users are advised to stop using the PAM key.

6.2.9 UPAM Macros

The following macros exist for UPAM:

- All service macros (e.g. OPEN, CLOSE, FCB)
- The **PAM** macro, which includes the following operations:

CHK	check the status of I/O processing
LOCK	lock a PAM block
LRD	lock a PAM block and read the contents into main memory
LRDWT	same as LRD; wait for I/O completion
RD/RDWT	read a PAM block into main memory and optionally wait for I/O completion
SETL	set the file pointer for further execution of a job
UNLOCK	unlock a PAM block
WRT/WRTWT	write data from main memory into a PAM block and optionally wait for I/O completion
WRTWU	write from main memory into a PAM block, wait for I/O completion, subsequently unlock the PAM block written
WT	wait for completion of an I/O operation

- If eventing is used in conjunction with UPAM, the following macros are available (the macros highlighted in bold type are mandatory; see also section 6.5.2, "Informing the User Job upon Termination of a UPAM I/O Operation"):

ENAEI	allocate an event item to a user job
DISEI	deallocate a user job from an event item
CHKEI	check the status of an event item
SOLSIG	send a request to an event item
ENACO	enable a contingency definition, i.e. allow it to control contingency tasks
DISCO	nullify a contingency definition, i.e. prohibit it from controlling contingency tasks
LEVCO	change the priority level of a contingency task or the basic task
RETCO	terminate a contingency task
CONTXT	provide read or write access to the register set and program counter (the "context") of an interrupted contingency task or the basic task

These macros are described in the "Executive Macros" manual.

FCB CREATE FILE CONTROL BLOCK ‹TYPE 0›

Operation	Operands
FCB	FCBTYPE=PAM [,LINK=linkname] [,IOAREA1={NO | relexp}] [,IOAREA2={NO | relexp}] [,EXIT={(relexp) | relexp}] [,BLKSIZE={STD | (STD,absexp)}] [,PAMREQS=absexp] [,PAMTOUT=absexp] [,SHARUPD={YES | NO | WEAK}] [,OPEN={INPUT | INOUT | OUTIN}] [,FILE=filename] [,PASS=password] [,RETPD=days] [,WRCHK={YES | NO}] [,OPTION={code | code,code)}]

See chapter 5 for a description of the operands.

Notes:

The RECFORM and RECSIZE operands can be specified, but are ignored by the UPAM action macro.

However, the BLKSIZE operand is required:

- if the operands IOAREA1 and/or IOAREA2 have not been specified in the FCB macro. In this case, the system reserves buffer areas. The buffer size corresponds to the value specified for BLKSIZE in the FCB macro.
- if the operands IOAREA1=relexp and/or IOAREA2=relexp have been specified in the FCB macro. In this case, the buffer sizes are checked when the file is opened.

In either case, the value of the LEN operand in the PAM macro must not exceed the value of the BLKSIZE operand in the FCB macro.

Exception:

If an I/O area is specified explicitly in the PAM macro (operand LOC= relexp), the BLKSIZE operand in the FCB macro is irrelevant.

PAM PERFORM UPAM ACTIONS (TYPE S)

All user requests to the DMS for UPAM actions are made using this macro.

Operation	Operands
PAM	fcbaddr [,{CHK, LOCK, LRD, LRDWT, RD, RDWT, SETL, UNLOCK, WRT, WRTWT, WRTWU, WT}] [,CHAIN=relexp] [,FECB=relexp] [,HP={absexp, +absexp, -absexp}] [,KEYFLD=relexp] [,LEN={STD, absexp}] [,LOC={1, 2, relexp}] [,MKEY={YES, NO}] [,REQNO=absexp]

fcbaddr — This operand specifies the address of the FCB (relexp) associated with the file.

CHK — This operand checks whether the specified I/O request has been completed. If so, the program proceeds. Otherwise, control is transferred to the address specified by the LOC operand in the user program.

LOCK — This operand Locks PAM blocks (see also the operand HP=).

LRD — As for LOCK; if locking was successful, processing continues as for RD.

LRDWT — As for LOCK and LRD; if locking was successful, processing continues as for RDWT.

RD — This operand initiates the reading of a data block from the file into main memory; the job continues immediately after the read operation has been initiated.

RDWT — As for RD, with the difference that the job does not continue until the read operation has been completed.

SETL — This operand causes the file pointer to be set to the PAM block specified.

UNLOCK This operand causes locked PAM blocks to be unlocked (see also the operand HP=).

WRT This operand initiates the writing of a data block from main memory into the file; immediate continuation of the job.

WRTWT As for WRT, with the difference that the job does not continue until the write operation has been completed.

WRTWU As for WRTWT; immediately after completion of the I/O operation, the PAM block just written is unlocked.

WT This operation causes the program to wait for the end of a particular request. The program continues upon completion of this request.

CHAIN= This operand specifies the symbolic address of the next element in a chain of PAM macros in list form. Such a chain must not contain more than 255 elements.

Default value:

The list is regarded as the last element in a chain.

FECB= This operand specifies the symbolic address of a File Event Control Block (see section 6.2.5). If this operand is specified, the REQNO operand must be omitted.

HP= This operand references a PAM block, namely:

- in the case of non-chained I/O, the PAM block to be transferred (or locked).
- in the case of chained I/O, the first in a series of PAM blocks to be transferred (or locked).

There are two methods of referencing a PAM block:

- Direct specification of the absolute logical block number (HP=absexp).
- Specification of a relative logical block number.

  HP={+absexp / -absexp}

  Relative logical block numbers relate to the file pointer.

  The absolute logical block number is computed as the sum of the file pointer and the specified relative block number.

Default value: HP = + 1 (i.e. sequential processing)

Programming notes:

- The first PAM block of a UPAM file is number 1. At the time a UPAM file is opened, the file pointer is set to an initial value of 0. This operand is ignored in WT or CHK operations. WT and CHK relate to I/O requests and not to PAM blocks.

- Any operation (except WT and CHK) not causing a branch to an error recovery routine has the effect that the file pointer is set to the number of the PAM block which is being accessed. This is also true of locking/unlocking even if SHARUPD=NO is specified. Normally, however, the file pointer remains unchanged if operations result in an error routine instead of being completed successfully.

- If n denotes the number of PAM blocks already allocated to a file and k is the value of the secondary allocation, the following rules must be observed:

  RD, RDWT, LRD, LRDWT — HP can define only a PAM block which has a number from 1 to n.

  WRT, WRTWT, WRTWU — HP can define only a PAM block which has a number from 1 to n+k. Whenever a PAM block of any number from n+1 to n+k is encountered, secondary allocation takes place automatically, the size of the file being extended to n+k PAM blocks.

  If a number of consecutive PAM blocks are to be written with one PAM macro (chained I/O), storage space for all blocks to be written must be allocated (at the latest) after a single secondary allocation.

  Thus, the following must apply:

  (block number of last block to be written) $\leq$ n+k

  LOCK, UNLOCK — HP must specify a logical block number greater than zero. This enables the locking and unlocking of PAM blocks which are currently not allocated. Such locking/unlocking does not lead to secondary allocation.

KEYFLD=

In the case of non-chained I/O, this operand specifies the address of a 16-byte area, into which the PAM key is placed during reading or from which it is taken during writing, as the case may be.

In the case of chained I/O and the processing of all associated PAM keys (operand MKEY=YES), this operand is mandatory and must contain the address of a sufficiently large area (number of PAM blocks to be transferred multiplied by 16 bytes).

If only the PAM key of the first of a series of PAM blocks is to be processed (MKEY=NO), the conditions stipulated above for non-chained I/O apply; in this case, the KEYFLD operand can optionally be omitted, whereupon the default value will take effect.

Default value:

For all read and write operations, usually the address of the ID1KEY1 field in the FCB. Exception: for the RDWT operation it is the address of the ID1KEY2 field in the FCB.

Programming notes:

- If a WT is issued for a successful read operation, either explicitly or implicitly, the 16-byte PAM key assigned to the block read is moved to this field. A CHK operation issued following the end of an I/O operation has the same effect as a WT operation.

- In a write operation, if a user places any information in the last 8 bytes of the 16-byte field defined by this operand, that information is written to the file as part of the PAM key which is assigned to the PAM block to be written (not recommended). UPAM generally creates the first 8 bytes of this field before the write operation begins.

- This operand is ignored by the WT, CHK, LOCK, UNLOCK and SETL operations.

LEN=

This operand specifies the length of the I/O buffer to be used.

The value of this operand can be in the range 1 through 32768.

Two cases should be distinguished:

- $1 \leq LEN \leq 2048$

  In this case, one PAM block is read or written with each PAM macro.

- $2049 \leq LEN \leq 32768$

  In this case, chained I/O is used.

  The number of PAM blocks to be transferred is determined from the value of the LEN operand as follows:

  If LEN is an integer multiple of 2048, the quotient of (LEN / 2048) gives the number of PAM blocks to be transferred.

  Otherwise, the quotient is rounded up to the next higher integer.

Notes:

- The LEN operand is ignored by the WT, CHK and SETL operations.

- See also section 6.2.9, FCB macro for UPAM, operand BLKSIZE.

LOC= This operand references the I/O buffer in main memory (though not in the case of CHK). The buffer must be capable of holding at least as many PAM blocks as are specified by the value of LEN. If the buffer is less than or equal to 4096 bytes (one page), it must be fully contained on one page and be aligned on a word boundary. If this area is longer than one page, it must:

1. begin on a page boundary
2. be virtually contiguous
3. not cross a segment boundary

=1 The buffer address is in FCB field IOAREA1.

=2 The buffer address is in FCB field IOAREA2.

=relexp This operand specifies the buffer address or, if a checked I/O request has not yet terminated, the continuation address.

Default value:

The IOAREA1 address in the FCB, provided this was not the last buffer address or the IOAREA2 address does not exist.

The IOAREA2 address in the FCB in all other cases.

Notes:

- In the case of the CHK operation, the LOC operand must not be omitted; in fact, the following specification is mandatory:

  LOC=relexp

- This operand is ignored by the operations WT, LOCK, UNLOCK and SETL.

- Wherever possible, the data buffers should be aligned on a 2 KB word boundary. They must be aligned if the address is greater than X'800'. This eliminates the need for intermediate data buffering under certain circumstances, and the resultant deterioration in performance.

MKEY= This operand is significant for chained I/O only.

=YES PAM keys are expected by the user for each PAM block read, or provided by the user for each PAM block written.

=<u>NO</u> Only the PAM key of the first of a series of PAM blocks is expected/provided by the user.

REQNO= This operand specifies the number of the I/O request assigned to this operation. If this operand is specified, the FECB operand must be omitted.

Default value: REQNO=1

Notes:

- This operand can have only values from 1 to n, where n is the value of the PAMREQS operand in the FCB.
- This operand is ignored by the LOCK, UNLOCK and SETL operations. For the WT and CHK operations, this is the way of specifying the I/O request for which the operation is to be carried out.
- Each request number can be assigned one asynchronous I/O operation only. The end of this asynchronous I/O operation is to be waited for: this can be specified either explicitly by supplying its request number in a WT operation, or implicitly by another read or write operation for the same request number. If an error is encountered during an implicit wait for the end of an I/O request, error code 997 appears, and the request which caused the implicit WT is not carried out.
- Request numbers are system resources; improper use may impair the efficiency of the system.

6.3 ISAM (INDEXED SEQUENTIAL ACCESS METHOD)

ISAM processes logical records in an indexed-sequential file. It may be used to:

- Create an indexed-sequential file in a sequential or non-sequential manner.
- Retrieve the logical records in the file in a sequential or non-sequential manner.
- Update records in a sequential or non-sequential manner.
- Insert new records in a file on the basis of their logical order in that file.
- Delete selected records from the file.
- Retrieve the next sequential record in a file which contains the requested flags.

Transfer of data to and from the file can be in either of two modes:

Move mode: The user specifies the address of the record in his program. The system is responsible for transferring data to or from the buffers.

Locate mode: The user requests the address of the current record in the buffer area. The user is responsible for transferring data to and from the buffers.

The following action macros are available in ISAM to control file processing:

ELIM Eliminate a record from the file.

GET Retrieve the next record in the file in ascending sequential order of the record keys.

GETFL Retrieve the next logical record that satisfies the flag criterion.

GETKY Retrieve the next record on the basis of a specified record key.

GETR Retrieve the next record in the file in descending sequential order of the record keys.

INSRT Insert a new record in the file in the position determined by its record key.

ISREQ Unlock a data block.

OSTAT Provide information about opened files (e.g. co-users).

PUT Add a logical record at the end of the file. PUT is normally used for the initial creation of a file, the logical records then being written in ascending sequential order of their record keys.

PUTX Write a record previously read by GET, GETR, GETFL or GETKY back into the buffer.

RETRY Retry an I/O operation, reposition a block or wait for it to be unlocked after the PGLOCK exit of the EXLST macro has been taken.

SETL Specify the position in the file at which subsequent processing is to begin. The beginning of the file, the end of the file, or the location of a record with the designated key can be specified.

STORE Place a new record in the file or replace an existing record; the position is determined by the record key.

6.3.1 ISAM File Structure

An ISAM file consists of data and index blocks. Data blocks are n*2048 bytes in length, where 1 ≤ n ≤ 16. Index blocks have a usable length of 2044 bytes.
The entries in the data blocks contain the user's logical records. Entries in the index blocks contain pointers either to the index blocks of the lower levels or, if it is the index block of the lowest level, to the data blocks. For each data block of the file there is, at the lowest level, an index entry of the length KL+4, where KL corresponds to key length (+ length of flags where applicable); the highest key contained in the data block is marked.

The index entries are always sorted in ascending order of the record keys.

The data blocks, also referred to as level-0 blocks, contain the logical records arranged in ascending sequence of the record keys.

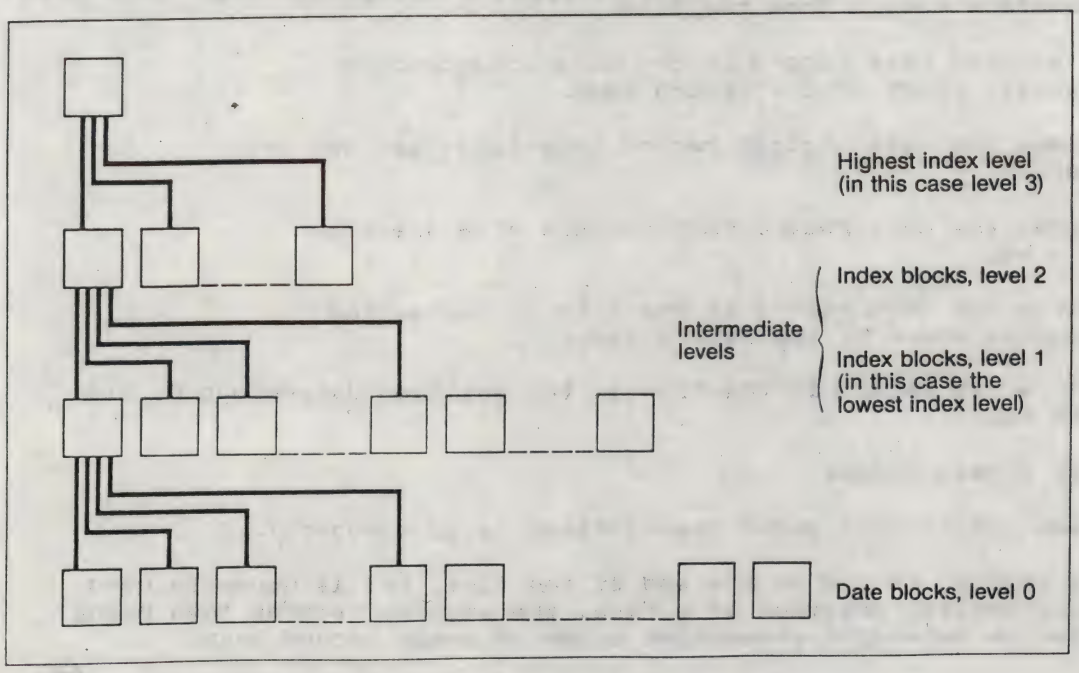


Fig. 6-8: ISAM file structure (in diagram form)

The highest level index block is always present in the file, even if it is a null file, i.e. contains no records. In addition to pointers, this index block contains 36 bytes of ISAM label information, which is used internally by ISAM. Therefore, in the calculations that follow, the usable length of the highest level index block is considered to be 2008 bytes rather than 2044 bytes.

In the case of smaller ISAM files, one index level is usually sufficient. In this case, the block of the highest index level is at level 1 and contains pointers to the data blocks.

Whenever a file grows so large that the highest level index block no longer has enough space to accommodate pointers to all of the blocks on the next lower level, that block is split in half and an index block is created, one level higher than the old one and containing two pointers, one to each half of the split block. The two blocks generated from the original index block are linked.

Notes:

- By linking index blocks of the same level, only 2044 bytes remain available as of Version 7.5, since the last 4 bytes in the block are reserved for linking.
  Existing files can be processed without any problem. The last four bytes are not used before processing, and then only until the next higher index level has been altered.

- Elimination of the PAM keys, which is required as a result of the new disk devices, necessitates a change in the format of ISAM files; in particular this means that the area available for records on each PAM page has been reduced.

  Recommendation: The maximum record length for ISAM files should not exceed "n*2000 bytes" when BLKSIZE=(STD,n) is specified. Use of records with a length of "n*2048 bytes" is still possible, but this will have an adverse effect on performance.

6.3.2 ISAM Record Format

ISAM supports two record formats, as follows:

F (fixed length)

V (variable length).

The block size can be n*2048 bytes, where $1 \leq n \leq 16$. The buffer size must correspond to the block size and must be a multiple (max. 16) of the standard size. The key must be contained at a fixed position within the record and be the same size for each record in the file. See section 6.1.3 for the format of logical records.

6.3.3 Opening an ISAM File

ISAM files can be opened in the following modes:

INPUT — To retrieve records from an existing file. The records may be retrieved sequentially or randomly.

OUTPUT — To create a new file.

EXTEND — To add records to the end of an existing file.

INOUT — To retrieve, delete, replace and insert records in an existing file.

OUTIN — To retrieve, delete, replace, insert and store records in a new file.

Table 6-4 summarizes the ISAM processing applicable to each OPEN mode.

Action macro \ OPEN mode	INPUT	OUTPUT	EXTEND	INOUT	OUTIN
GET	B			B	B
GETR	B			B	B
GETFL	B			B	B
GETKY	B			B	B
PUT		B	B	B	B
PUTX				B	B
INSRT				M	M
STORE				M	M
ELIM				X	X
SETL	X			X	X

Table 6-4: ISAM action macros in relation to OPEN modes

Where:

M action macro: functions in move mode (also allowed in locate mode if the work area was supplied with data)
B action macro: functions allowed in move or locate mode
X action macro allowed

If there is no entry in the table, the action is not allowed.

Overlap buffering of I/O operations is supported for all OPEN modes. Buffering is optional for the GET and GETR action macros.

These macros do not require the supervisor call instruction (SVC) until all the logical records in the current buffer have been processed.

If this look-ahead buffering is not desired, the programmmer should specifically omit the OVERLAP operand from the FCB. If buffering is performed and the program switches from forward to reverse processing, or from sequential to random processing, the I/O operation which is to ready the next sequential buffer is initiated immediately. This can result in an uncompleted operation because, if processing was exclusively sequential, the system supplies data which is not accessed.

PUT operations are always overlapped.

If the padding factor (PAD=) is specified, space is reserved only in the case of sequential creation of the file via the PUT action macro. The INSRT and STORE action macros attempt to use all the space available in a data block.

While a file is open, the KEYLEN and KEYPOS fields in the P1FCB have a value of n-1. If the file contains fixed-length records (RECFORM=F), the value of the KEYPOS field is (n+4)- 1 = n+3. In both cases the value of the KEYPOS field is contained in either the catalog entry or the FCB macro or FILE command.

Note:

When opening a new ISAM file a new value is calculated for the BLKSIZE field in the P1FCB. The PAD value from the FCB is taken into account. The value calculated (specified in BLKSIZE - PAD) is then stored in the BLKSIZE field.
This does not apply to INPUT processing.

Example:

```
Value      BLKSIZE=(STD,3)
           PAD=15 (default value)
```

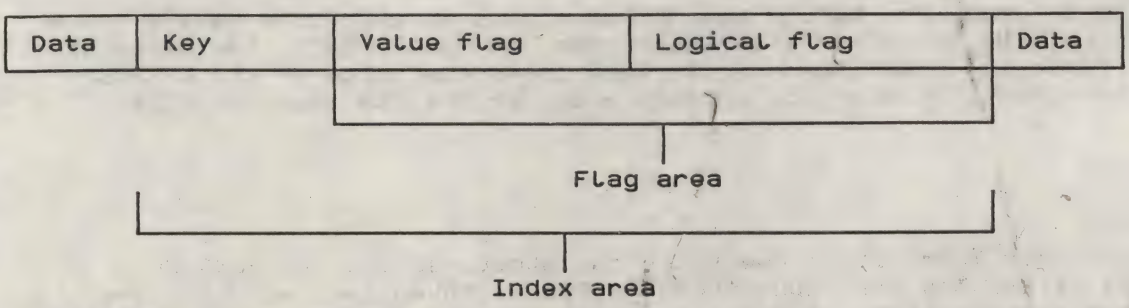
The value of the BLKSIZE field during processing is X'1467'.

6.3.4 Flagged ISAM Files

Flags enable the user to make use both of the ISAM key and of other record attributes in order to qualify and accelerate access. There are two types of flag: value flags and logical flags. Flagged ISAM files may contain either value flags, logical flags or both.

The flag is part of the user's logical record; the value flag immediately follows the key, and the logical flag immediately follows the value flag (or the key, if there is no value flag).

User record



The only restriction on key length, value length and logical length is that their total length must not exceed 255 and the key length must be at least 1.

Flag information is propagated throughout all index levels of the ISAM file.

Value information in the index entries is created on the basis of the value specified by the user in the VALPROP=operand.

If the MAX (MIN) function is specified, each index entry will contain the highest (lowest) value flag of the next lower level block (either data or index block) to which it points. Whether propagation of the maximum or minimum value flag has any point depends on the way in which the records are subsequently to be made available. The usefulness of a value flag is all the greater the more evenly the value flags rise or fall with an ascending ISAM key (see example below).

<u>Opening a flagged ISAM file</u>:

If either VALLEN or LOGLEN is not equal to zero, the file is assumed to be flagged. If the file already exists (the OPEN mode is INPUT, INOUT or EXTEND), the catalog values of VALPROP or VALLEN are used. LOGLEN must always be specified when the file is created with a flag area. The catalog values cannot be changed by the FCB macro or FILE command.

Example:

Personnel file, value flag = year-of-birth, VALPROP=MIN

Thus:

From each data block, the earliest year-of-birth value for the employees contained therein is transferred to the index block.

1. Suitable search query:

   Retrieve all records for employees born **before** 1950.

2. Unsuitable search query:

   Retrieve all records for employees born **after** 1950.

In the first case, whole data blocks can be skipped immediately because of the value flag.

In the second case, every record in each block must be checked. Here a file with VALPROP=MAX would be advisable.

In the logical flag it is possible to code, in bit form, dual attributes of the object described in the data record. All logical bytes of a data block (or index block) are then linked by means of logical OR and passed on to the (next higher) index entry. In searching for records with particular attributes, the DMS can then decide, on the basis of knowing the index entry, whether records with these attributes are present in the referenced block.

6.3.5 ISAM Shared File Update

Under ISAM a file can be updated by several users at the same time. This support is based on two principles:

Shared file update is defined by the user at file OPEN time. To avoid conflicts which may arise if several programs attempt to access and update the same data block, a mechanism is provided to allow the user to lock/unlock data blocks. If the lock/unlock function has been omitted, and it is necessary for the data block to be locked, this function is carried out automatically by ISAM. The following paragraphs describe in detail the shared update facilities available to the user.

Opening an ISAM file

The first user to open an ISAM file can open it with any combination of the values permitted for OPEN and SHARUPD. The following table indicates which OPEN/SHARUPD combination will be permitted when user B tries to open the file which user A has already opened. If more than one user has already opened the file, the OPEN/SHARUPD combination of user B will be compared with the OPEN/SHARUPD combination of every user who has opened the file; user B can open the file only if each such comparison permits him to do so.

			USER B									
			SHARUPD=YES					SHARUPD=NO				
			INPUT	INOUT	EXTEND	OUTIN	OUTPUT	INPUT	INOUT	EXTEND	OUTIN	OUTPUT
USER A	SHARUPD=YES	INPUT	X	X	X			X				
		INOUT	X	X	X							
		EXTEND	X	X								
		OUTIN	X	X								
		OUTPUT										
	SHARUPD=NO	INPUT	X					X				
		INOUT										
		EXTEND										
		OUTIN										
		OUTPUT										

Table 6-5: Processing an ISAM file in shared update mode

ISAM action macros

If a file is updated by more than one user, it is absolutely mandatory for parts of the file to be locked, so as to prevent more than one user from accessing and updating the data block at the same time. This blocking is performed at data block level. Some of the ISAM action macros (e.g PUT, STORE, INSRT, ELIM (with KEY)) already contain all 3 functions; i.e. the macro locks, updates and unlocks the data block concerned.
With the other macros, two macros are required to carry out these functions:

- GET, GETR, GETKY or GETFL macro to lock the block.
- PUTX or ELIM (without KEY) macro to update and unlock the block.

An optional LOCK/NOLOCK operand may be specified for the GET, GETR, GETKY or GETFL macro.

If a data block has been locked by GET, GETR, GETKY or GETFL with the LOCK specification, other users can read the block (e.g. with SETL or with GET, GETR, GETKY or GETFL with the NOLOCK option), but they cannot update it; i.e. they cannot lock it by specifying LOCK in a macro. If NOLOCK is specified, other users can read, lock, update and unlock the data block.

If SHARUPD=NO is specified, the LOCK operand is ignored in the GET, GETR, GETKY or GETFL macro; the data block is not locked.

Note:

In this system version, registers 0, 1, 14 and 15 are not destroyed by all of the action macros. The user should not, however, assume that this is always the case, since registers 0, 1, 14 and 15 will be destroyed by every action macro in all future subsequent versions.

Use of the LOCK/NOLOCK operand with SHARUPD=YES

A data block which has been locked by one user can be read but not updated by other users. Furthermore, it is not possible for two users to lock the same block simultaneously.

On the other hand, a user who is reading a data block but has not locked it permits other users to read and update the block.

Consequently, the use of NOLOCK in the GET, GETR, GETKY and GETFL macros can be seen as extending the degree of data block shareability. However, LOCK must be specified if the data block is to be updated via a PUTX or ELIM macro (without KEY) or if the user wants to ensure that no other user will update the data block while he is reading it.

Locking data blocks

A user can have more than one ISAM file opened with SHARUPD=YES but he cannot lock more than one data block at a time.

The locking of data blocks is conceived as follows: the first step undertaken by every ISAM action macro which is to lock a data block is to unlock any data block this user has locked in any file. Moreover, every ISAM action macro will unlock any locked data block in the file for which the macro is issued. However, GET, GETR and PUT will not actually unlock the data block if the same data block is to be relocked later in the macro processing; PUTX and ELIM (without KEY) do not unlock the data block until updating has been completed.

Implicit locking and unlocking of data blocks

Data blocks can be explicitly locked by the GET, GETR, GETKY and GETFL macros and unlocked by PUTX and ELIM (without KEY). In addition, however, it is possible for every action macro to be able to lock/unlock certain data blocks implicitly.

If SHARUPD=YES is specified, the appropriate data block is locked before it is updated by STORE, INSRT or ELIM (with KEY) and then unlocked again.

In order to avoid deadlock situations, each user can lock only <u>one</u> data block, regardless of how many files he has opened. This is ensured in the following way: whenever the user issues an action macro to lock a data block, the data block (of any file) previously locked by the user is unlocked. This happens, for example, if the file is opened with SHARUPD=YES and the macros STORE, INSRT and ELIM (without KEY) have been issued, or the macros PUT, GET, GETR, GETKY and GETFL with the LOCK specification were used.

If a PUT macro is issued for a file opened in locate mode, IOREG is updated and the appropriate data block is locked.
The user must move the record to the location defined by IOREG (MVC).
The block is not unlocked until it is complete, or until another action macro has been issued.

If a record is modified, rewriting takes place with PUTX. However, the block is not stored until an action macro is issued to access another block. Until this occurs, the block remains locked.

Updating a data block

Before a data block can be updated, it must be locked.
When the update is complete, a check is made as to whether the index blocks have to be updated. If this is the case, the index block is locked and modified. Index and data blocks are unlocked once they have been updated and the next higher user level has been locked.

Error exits

LOCK exit:

If the OPEN/SHARUPD entries make it impossible to open the file, or if problems occur when opening a file already opened by other users, control is transferred to the user at the LOCK exit.

USERERR exit:

This exit is taken when action macros are wrongly used (e.g. PUTX without a preceding GETKY).

The user may also be given control at this exit if he has issued a GET, GETR or GETFL macro after the PGLOCK exit was taken without first repositioning the internal pointer (cf. the description of the PGLOCK exit).

PGLOCK exit:

This exit is meaningful only if this file was opened with SHARUPD=YES. A branch is made to the PGLOCK exit if the program tries to lock a block that has already been locked.

If this exit is not used (default value or NO is specified) and the required block is locked, the user waits until he can access this block. He is not informed of the "busy" status.

Control is passed to the user at this exit whenever a data block has to be referenced during processing of an action macro and is inaccessible because of the manner in which another user is locking it. (For example, if user A has locked a data block and user B also tries to lock it, user B will be given control at the PGLOCK exit.)

It is possible for any ISAM macro issued for a file opened with SHARUPD=YES to cause control to be passed to this exit.
If this exit is taken, the "internal pointers" are in most cases invalid. Consequently it is necessary to reposition the pointers: the RETRY macro repeats the operation which caused the PGLOCK exit to be taken, or repositions the appropriate pointer (cf. the description of the RETRY macro). Macros for direct access can also be issued (e.g. GETKY, STORE, INSRT, SETL and ELIM (with KEY)).

Notes on using shared update facilities

- When to use ISAM shared update

  Specifying SHARUPD=YES for a file causes ISAM system tables to be set up in class-4 memory, while SHARUPD=NO causes the creation of at most two table entries.

  If SHARUPD=YES is specified, these tables are updated with every ISAM macro. If SHARUPD=NO is specified, the entries are updated at OPEN/CLOSE time only.

  Thus, in order to save time and class-4 memory, SHARUPD=YES should be specified only when it is absolutely necessary for more than one user to open the file at the same time, and when at least one of the users wishes to update it (e.g. OPEN=INOUT, OUTIN, EXTEND).

- Opening and closing an ISAM shared update file

  When a file is opened with SHARUPD=YES, detailed tables are created in class-4 memory. These tables are checked and maintained by OPEN.

  In general, to avoid unnecessary overhead, a user should not open a file until he is ready to process it, and he should not close it until processing is finished.

  CLOSE processing will unlock any data block the user has locked in that file.

- Locking and unlocking data blocks

  Issuing macros which lock or unlock data blocks can cause unnecessary overhead.

  In some situations, this overhead is unavoidable, as in the case of the GET/PUTX sequence, where SHARUPD=YES is specified and the GET macro has to lock the data block and the PUTX macro has to unlock it.

  However, by dint of careful programming, the user can avoid such situations, which arise when GET macros are issued either with the LOCK specification for two or more files, or with the LOCK and UNLOCK specifications for the same file.

- Use of records with the same key for shared update files

  If a user is reading a file sequentially (i.e. is using a GET, GETR or GETFL macro) and the file was opened with SHARUPD=YES and DUPEKY=YES, he must be aware that the file may contain records with the same key. If user B has modified an index page associated with the data block user A wants to read, it may be necessary for ISAM to reposition the user in the file on the basis of the modified index pages. To do this, ISAM issues a SETL KEY to reposition the user. However, if the key involved is one of a sequence of duplicate keys, the user may be positioned to an undefined record in the sequence. Therefore, it is the user's responsibility to keep track of which record in the sequence he has processed.

  The RETRY macro may cause the same unexpected positioning for a file containing duplicate keys. This condition is described in detail under the section referring to the RETRY macro.

6.3.6 General ISAM Programming Notes

If a file was opened in the OPEN mode OUTPUT or OUTIN and then closed again, a null file is created. The number of the last page (LASTPG) is then incremented by one, e.g. when BLKSIZE=(STD,6), LASTPG=7.
If a null file is created by deleting all records using the ELIM macro, the last page pointer is retained.

ISAM does not prohibit opening of a null file if the OPEN mode is other than INPUT. If an empty file is opened in INPUT mode, control is passed to the user at the OPENER address of the EXLST macro. If, when processing a null file, an action macro contingent on the existence of data records is executed (e.g. GET), an EOF exit is taken.

When creating a file with BLKSIZE=(STD,n) where n is greater than 1, the user is required to allocate n+1 PAM pages with /FILE prior to opening the file. Failure to allocate sufficient space will result in an OPENER exit being taken.

The processing time required to insert new records is reduced if the user can roughly estimate the extent of future file expansion. This estimate is specified via the PAD operand when creating the file. Although the ISAM routines attempt to optimize the splitting of data blocks when new records are entered in the file, such splitting requires a number of additional I/O operations and should therefore be avoided.

When determining the PAD value for a file the following should be noted: Should the user select a PAD value which, added to the RECSIZE, exceeds the BLKSIZE, the DMS will prohibit file opening. Error code 0DBC is used to indicate this error condition.
If action macros are used, each of them operates with an internal pointer. Usually, this pointer is set to the record currently being processed by the action macro. The pointer is modified by the next action macro. It may reference an "imaginary" position.

The ISAM pointer conventions are discussed at the end of this section.

Note for ISAM processing when using default functions

For example, if an FCB macro is specified as follows:

FCB LINK=X,EXIT=Y,IOREG=9

the file will be an ISAM file with RECFORM=V and KEYPOS=5.

These default values are assigned at assembly time (macro expansion time).

Therefore, if a user issues the following FILE command:

/FILE␣A,LINK=X,RECFORM=F,OPEN=OUTPUT

the RECFORM will in fact be changed to "fixed", but KEYPOS will remain 5.

A second example uses the same FCB and the command:

/FILE␣B,LINK=X,OPEN=OUTPUT

Since IOREG is specified in the FCB macro, the ISAM macros specified for this file, when open, will operate in locate mode.

As no value is specified for RECSIZE, RECSIZE=BLKSIZE is assumed. As PAD is also not specified, 15% of the buffer size is reserved for padding.

If the user now attempts to issue a PUT macro (in locate mode) without changing RECSIZE, RECSIZE will be greater than the area available in the buffer for the user's records, and the PUT macro will result in an error.

Size of ISAM files

Because of the many options which the user can exercise in the creation of an ISAM file, it is generally not easy for him to determine the size of the file, the number of index blocks or the number of index levels. The following formulae facilitate the specification of certain values, but they should not be regarded as anything more than rules of thumb.

- **Data blocks:**

RDB=FLOOR (BS/RS)
NDB=CEIL (NR/RDB)

where:

FLOOR = Greatest integer less than
CEIL = Smallest integer greater than
RDB = Records per data block
BS = Block size

a) Block size = 2048 times the number of PAM blocks specified in the BLKSIZE operand.

b) If the file was created sequentially by means of the PUT macro, BLKSIZE must be adjusted accordingly by being multiplied by (100-PAD)/100.

Note:

The default value for PAD is 15.

c) If the file was created either sequentially or randomly by means of the STORE or INSRT action macro, the file is densely packed. Therefore the value of BLKSIZE given in a) above should be used regardless of any PAD value.

RS = Record size

a) For fixed-length records (RECFORM=F), add 4 to the value specified in RECSIZE.

b) For variable-length records, use the average record length.

Note:

As a result of support provided for new disk devices, there may be changes with respect to the processing of PAM pages. The area within PAM pages which can be used as the data area may be reduced as of the next version of the system.

- **Index blocks:**

$$NB_i = CEIL\left(\frac{NB_{i-1} * (KL+4)}{size}\right)$$

where:

NB_i = Number of blocks at level i.

NB_{i-1} = Number of blocks at level i-1 (NB_0 = NDB).

KL = Key/flag length, i.e. the sum of:

KEYLEN + LOGLEN + VALLEN

size = The effective size of the index block at level i.

a) If level i is the highest level, size = 2012 bytes, otherwise size = 2044 bytes.

b) If the file is created sequentially by the PUT, STORE or INSRT action macro, the size given in a) above must be adjusted by multiplying it by 0.5, except when i is the highest level index.

c) If the file is created completely randomly, the value given in a) above is only approximately accurate.

d) If the file is created randomly by means of sequential PUT, STORE or INSRT macros, an adjustment factor between 0.5 and 1.0 should be used.

Example:

Given:

STD blocks, 1000 80-byte records, KEYLEN=8, LOGLEN=2, VALLEN=2, RECFORM=F, PAD=10.

The file was created sequentially with PUT action macros.

RDB = FLOOR (2044*0,9/(80+4))=21

Each data block therefore contains 21 records.

NDB = CEIL (1000/21) = 48

There are 48 data blocks in the file.

$$NB_i = CEIL\left(\frac{48*(8+4)}{2012}\right) = 1$$

Thus only one index block is required; the total number of halfpages in the file is 49.

Given:

STD blocks, 100000 100-byte records (average), KEYLEN=5, LOGLEN=1, VALLEN=0, RECFORM=V.

The file was created sequentially with the STORE action macro.

RDB = FLOOR (2044/100) = 20

Each data block therefore contains approximately 20 records.

NDB = CEIL (100000/20) = 5000

There are approximately 5000 data blocks in the file.

$$NB_1 = CEIL\left(\frac{5000*(6+4)}{2048/2}\right) = 49$$

Thus, there are 49 level-1 index blocks in the file.

$$NB_2 = CEIL\left(\frac{49*(6+4)}{2012}\right) = 1$$

The highest level index block in the file is therefore level 2. It can be seen from the above calculation that the file would have to contain about 4 times the specified number of data records before index level 3 is required. The total number of halfpages required by the file would be:

5000 + 49 + 1 = 5050

Data block splitting

If a new record does not fit where the index indicates it should, the data block is "split" in such a way that the two halves of the data block are moved from the original block into new, empty blocks.

Example:

Splitting a data block. Assume the following file exists:

Block 1 - Index block entry 1

Key 7 Block 2	Key 20 Block 3	Unused

Block 2 - Data block

Record with key 1	Record with key 3	Record with key 7

Block 3 - Data block

Record with key 10	Record with key 15	Record with key 20

The user now specifies that a record with a key of 5 is to be inserted. Since no space remains in block 2, which is the logical block that should contain the record, block 2 is split and the file now has the following structure:

Block 1 - Index block

Key 3 Block 4	Key 7 Block 5	Key 20 Block 3	Unused

Block 2 - Unused block

Unused

Block 3 - Data block

Record with key 10	Record with key 15	Record with key 20

Block 4 - Data block

Record with key 1	Record with key 3	Unused

Block 5

Record with key 5	Record with key 7	Unused

Note:

When splitting a data block two new PAM pages are used. The old page is released for future use.

6.3.7 ISAM Pointer Rules

ISAM action macros (including OPEN) perform two main functions:

1. Setting an internal pointer to a specific record.
2. Executing the desired action for the record currently being referenced.

Notes:

- A pointer is updated before the action is performed.
- Certain macros do not require the pointer to be updated.
- With certain macros, only the pointer is updated and no other action is performed.

The ISAM macros are summarized in the following table.

ISAM action macro	Pointer	Action	Comments
ELIM	If KEY is specified, the pointer is moved to the first record with the specified key. Otherwise the pointer is not moved.	Eliminates the record from the file.	ELIM can be regarded as a leftshift operation of that portion of the file to the right of the defined record. Consequently, successful ELIMs need not update the pointer.
GET	The pointer is moved forward one record.	Retrieves the record referenced.	1. If the pointer refers to a record outside the file, the user is given control at his EOFADDR address. 2. If the previous macro was a SETL or ELIM for a record with a specified pointer, the pointer is not moved before the record is retrieved.
GETFL	The pointer is moved to the record retrieved. If no record is is retrieved, the file is positioned so that a subsequent GET (GETR) will either retrieve the record with a key greater than (less than) or equal to the key within the defined limits, or result in an EOF condition.	Retrieves the next sequential record which satisfies the flag criteria.	The record which would have been retrieved by a corresponding GET or GETR macro is the first record scanned for the flag criteria when a GETFL macro is issued.
GETKY	The pointer is moved to the record with the specified key, or to the position in the file where such a record would exist, if a matching key is not found.	Retrieves the record referenced, if it exists.	In terms of pointer theory one can consider a GETKY for a non-existent record to be equivalent to a GETKY for a record of length 0 in between two existing records. Thus, a subsequent GET or GETR will work correctly (see GET, comment no. 1).

ISAM action macro	Pointer	Action	Comments
GETR	The pointer is moved backward one record.	Retrieves the record referenced.	1. If the pointer points to a record situated before the beginning of the file, the user is given control at his EOFADDR address. 2. See GET, comment no. 2.
INSRT	The pointer is moved to the position specified by the record key.	Inserts the record at the location pointed to.	The record is not inserted if a record already exists with the given key. However, the pointer still points to the place in the file at which the record would have been inserted. This record can be described as a record of length 0. A GET macro following an unsuccessful INSRT macro retrieves the duplicate record.
OPEN	The pointer is positioned to an imaginary record immediately before the first record in the file.		OPEN is decribed here only in relation to its effect on the pointer. The action caused by OPEN is described elsewhere.
PUT	The pointer is moved just beyond the current end of the file, if it is not already there.	Places the record in the file at the referenced position.	
PUTX	The pointer is not moved.	Places the record in the file at the referenced position.	An additional check is made to ensure that a successful GET, GETR, GETFL or GETKY macro has been issued just prior to PUTX. Thus, the fact that the pointer is not updated cannot cause inadvertent errors.

ISAM action macro	Pointer	Action	Comments
RETRY	Depends on the operation to be repeated, or on the positioning operation.	Performs the last action macro, positions the file for the user program or places the program in a queue.	RETRY itself does not affect the pointer except when it results in the execution of an action macro which does move the pointer.
SETL	The pointer is moved to: 1. an imaginary record immediately before the first record in the file (if B is specified). 2. an imaginary record immediately after the last record in the file (if E is specified). 3. the (first) record with the specified key or the place in the file where such a record would exist.	No action takes place.	1. Using SETL effectively nullifies the pointer movement associated with a subsequent GET or GETR macro. 2. For the purpose of clarifying the pointer placement, a SETL macro (for a subsequent GET or GETR macro) for a non-existent record may be considered to be a SETL macro for a record of the length 0, albeit at the correct position. This does not cancel updating of the pointer by the subsequent GET or GETR macro.
STORE	The pointer is moved to the position specified by the record key.	Stores the record at the desired location.	If DUPEKY=YES: If a record with the given key already exists in the file, the pointer is set to the position immediately following the duplicate record. The record is then stored.

Table 6-6: Summary of rules for ISAM pointers in action macros

6.3.8 Use of PAM Keys in ISAM

The rightmost eight bytes of the PAM key in ISAM files are:

LLPPUUFF

where:

LL is the page length (i.e. the highest byte position used in the page).

PP is a pointer to the first logical record in a data block. If this is an index page, the value is 0; it will also always be 0 for data pages as of Version 7.1.

UU contains the location of the last record to be used.

FF is a pointer to the first block of free space in the page. If there is no free space, FF points to the end of the block.

ISAM records within a data block are chained; record i points to record i+1; the last record points to record 1.

6.3.9 ISAM Macros

FCB CREATE FILE CONTROL BLOCK (TYPE 0)

The following format indicates the operands which may be used in the FCB macro in conjunction with ISAM. These operands are described in detail in section 5.4, "Service Macros for Disk Files".

Operation	Operands
FCB	FCBTYPE=ISAM [,BLKSIZE={STD | (STD,absexp)}] [,DUPEKY=YES] [,EXIT={(relexp) | relexp}] [,FILE=filename] [,FORM=SHORT] [,IOREG=regno] [,KEYARG=relexp [,KEYLEN=absexp][,KEYPOS=absexp]] [,LINK=linkname] [,OPEN={INPUT | OUTPUT | EXTEND | INOUT | OUTIN}] [,OVERLAP=YES] [,PAD=absexp] [,PASS=password] [,RECFORM={{F | V} | ({F | V} [,{A | M | N}])}] [,RECSIZE=absexp] [,RETPD=days] [,SHARUPD={YES | NO}] [,LOGLEN=absexp][,VALLEN=absexp] [,VALPROP={MAX | MIN}] [,WROUT={NO | YES}] [,WRCHK={YES | NO}] [,OPTION={code | (code,code)}]

Note:

Table 5-2 in section 5.4.4 summarizes the operands which may be specified. DUPEKY=NO is applicable only if neither FCB nor FILE contains the operand DUPEKY=YES.

If OPEN=INPUT, INOUT or EXTEND, KEYLEN is always overwritten with the information contained in the ISAM label.

ISAM ACTION MACROS

ELIM ELIMINATE RECORD (TYPE R)

The ELIM macro eliminates records from a file.

Operation	Operands
ELIM	$\left\{\begin{matrix}\text{fcbaddr}\\ \\ (1)\end{matrix}\right\}[,\left\{\begin{matrix}\text{KEY}\\ \\ (0)\end{matrix}\right\}]$

fcbaddr — This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) — The FCB address is located in register 1.

KEY — The key of the record to be eliminated is located at the address defined by KEYARG in the FCB.

(0)
- The address in register 0 is different from that in the FCB. Here it refers to the record to be eliminated.
- The address in register 0 is the same as that in the FCB. Here the last record previously retrieved by a GET, GETFL, GETKY or GETR macro is eliminated.
  The latter is also the default value if the operand [KEY/(0)] is not specified.

Programming notes:

- No other ISAM action macro can be executed between the associated GET, GETR or GETKY macro and the action macro ELIM (without KEY).
- If the KEY operand is specified and a record with the specified key cannot be found, control is returned to the user at the NOFIND exit (see the EXLST macro).
- If duplicate keys exist in the file and if the KEY operand is specified, the record first placed in the file is eliminated.
- Although the ELIM macro (without KEY) must normally be preceded by a GET, GETR, GETFL or GETKY macro, it may be executed for the first record in the file if it follows an OPEN macro in INOUT mode.

GET GET RECORD (TYPE R)

The GET macro retrieves the next sequential record of a file.

Operation	Operands
GET	{fcbaddr / (1)} [, {area / (0)}] [, {LOCK / NOLOCK}]

fcbaddr — This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) — The FCB address is located in register 1.

area — This operand specifies the address of the area into which the record is moved (exit address).

(0) — The address of the area into which the record is to be moved is located in register 0.

LOCK — This operand indicates that the data block containing this record is to be locked once the record has been retrieved.

NOLOCK — This operand indicates that the data block containing this record is not to be locked following execution of this macro.

Programming notes:

- The "area" operand is ignored if the IOREG operand was specified in the FCB.
- If a record beyond the end of the file is requested, the user is given control via EOFADDR (see the EXLST macro).
- If the file contains records with duplicate keys, they will be retrieved on a "first in, first out" (FIFO) basis.

GETFL GET RECORD BY FLAG (TYPE S)

The GETFL macro is used to retrieve the next record (in forward or reverse direction) within the specified range.
Its flag must satisfy the conditions specified by the macro.

Operation	Operands
GETFL	{fcbaddr | (1)} [,{area | (0)}] [,{LOCK | NOLOCK}] [,LIMIT={KEY | END}] [,LOGTEST={ANY | ALL}] [,MF=L] [,REVERSE=YES] [,VALTEST={GT | GE | EQ | NE | LE | LT}]

fcbaddr — This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) — The FCB address is located in register 1.

area — This operand specifies the address of the area into which the record is to be moved (exit address).

(0) — The address of the area into which the record is to be moved is in register 0.

LOCK — The data block containing this record is to be locked before the record is read.

NOLOCK — The data block containing this record is not to be locked following execution of this macro.

LIMIT= — The file scan for the next record starts from the current position in the file. The positioning may be the result of a SETL or GETKY macro, for instance.

=KEY — The scan continues until:

a) the last record which (in forward scan) has a key less than the key contained in the key field of the area referenced by KEYARG.

b) the last record which (in reverse scan) has a key greater than the key contained in the key field of the area referenced by KEYARG.

=END — The scan continues until the end of the file is reached.

LOGTEST=ANY — If this operand is specified, only those records are retrieved whose logical flag matches at least one of the bits of the mask stored in the logical flag portion of the area referenced by KEYARG.

=ALL — A record is retrieved if its LOG field bit configuration coincides with the bit configuration in the mask, although additional bits may be set in the record LOG field.

MF= — This operand specifies the type of macro expansion. If no entry is made, type R is assumed.
If L is specified, the following two-word operand list is generated:

Word 1	
Operand byte 1 (1 byte)	FCB address or register number (3 bytes)
Word 2	
Operand byte 2 (1 byte)	Area address or register number (3 bytes)

The meaning of the fields in the operand list is as follows:

Operand byte 1

- Coded GETFL operands (see Table 6-7a).

FCB address/register

- Either the FCB address or a register (right-justified) containing the address of the FCB.

Operand byte 2

- Coded GETFL operands (see Table 6-7b).

Area address/register

- The address of the area to which the record is to be moved, or a register (right-justified) containing the address of that area.

Bit position				Meaning (VALTEST=)
7	6	5	4	
0	0	1	0	GT
0	1	0	0	LT
0	1	1	1	NE
1	0	0	0	EQ
1	0	1	0	GE
1	1	0	0	LE
0	0	0	0	null or invalid
3				1 LOGTEST operand specified 0 LOGTEST null or invalid
2				1 LOGTEST=ALL 0 LOGTEST=ANY
1				1 LIMIT=KEY 0 LIMIT=END, null or invalid
0				1 REVERSE=YES 0 REVERSE=null or invalid

Table 6-7a: GETFL operand bit configuration - byte 1

Bit position	Meaning
7	1 LOCK specified (or default value) 0 NOLOCK specified
6	1 FCB address in register 0 FCB address specified
5	1 Area not specified 0 Area specified
4	1 Area address in register 0 Area address specified
3-0	not used

Table 6-7b: GETFL operand bit configuration - byte 2

A GETFL macro which is to use operands contained in a separate operand list has the following format:

Operation	Operands
GETFL	MF=(E,{list / (1)})

If this format is used, the code is generated to load registers 0 and 1 (based on the contents of the operand list), and the GETFL SVC is then issued.

The "list" operand specifies the location of the operand list to be used for this action (exit address). If the operand list shows that the FCB and area addresses are contained in registers, these registers must be loaded before the macro is executed.

Programming notes:

- The "area" operand is ignored if the IOREG operand was specified in the FCB (assuming locate mode).
- If a record whose flag is within the specified limits is not found, control is passed to the NOFIND exit (see the EXLST macro).
- If the file is positioned to a record containing the LIMIT key specified in a GETFL action macro, the NOFIND exit is taken and the file position is left unchanged.
- If LIMIT=END is specified or taken as the default value, file scanning proceeds to the end of the file if processing is in forward mode, or to the beginning of the file if in reverse mode. If file search conditions are not satisfied, the EOF exit is taken.
- If VALTEST and LOGTEST are both specified, both conditions must be met for the record to be retrieved.
- If neither VALTEST nor LOGTEST is specified, all the records are searched until the limit is reached. If a KEY limit was specified, the NOFIND exit is taken; if LIMIT=END is specified, the EOF exit is taken.
- If the KEYARG of a GETFL macro references a logic mask containing only binary zeros, the user error exit is taken. If LIMIT=KEY is specified, the file is processed in forward/reverse direction and the key portion of the area referenced by KEYARG is less/greater than the key of the record at which the file is currently positioned, control is transferred to the USERERR address via the EXLST macro.

- Any attempt to apply the GETFL macro to a file created without flags, and any attempt to output a record to a flagged file which is not large enough to accommodate a full index (key, value flag, and logical flag) will result in control passing to the USERERR address. For flagged files, KEYARG must point to a user-defined field large enough to accommodate the entire index (KEY, value flag and logical flag).

REVERSE=YES
: The scan takes place in reverse order (toward the beginning of the file).

VALTEST=
: Only those records are retrieved whose value flag is related as required to the information stored in the value flag portion of the area referenced by KEYARG.

Meaning of the relationship symbols:

=GT
: greater than

=GE
: greater than or equal to

=EQ
: equal to

=NE
: not equal to

=LE
: less than or equal to

=LT
: less than

GETKY GET RECORD WITH SPECIFIED KEY (TYPE R)

The GETKY macro retrieves, in random access, a record with the specified key. Prior to execution of the macro, the record key must be stored at the address specified in the KEYARG operand of the FCB.

Operation	Operands
GETKY	{fcbaddr | (1)} [, {area | (0)}] [, {LOCK | NOLOCK}]

fcbaddr — This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) — The FCB address is located in register 1.

area — This operand specifies the address of the area into which the record is to be moved (exit address).

(0) — The address of the area into which the record is to be moved is in register 0.

LOCK — The PAM block containing the record with the specified key is to be locked once the record has been retrieved.

NOLOCK — The PAM block containing the record with the specified key is not to be locked once the record has been retrieved.

Default value: Move mode, i.e. the record is transferred to the address "area".

Locate mode: This macro places the address of the record in the register specified by the IOREG operand.

Programming notes:

- The "area" operand is ignored if the IOREG operand was specified in the FCB.
- If a record with the specified key is not found within the file, the user is given control at the NOFIND address (see the EXLST macro).
- After GETKY is issued, the position indicators for GET and GETR are set as if a GET macro had been issued to retrieve a record. If the GETKY macro was not successful, a subsequent GET or GETR macro is used to retrieve a key greater or less, respectively, than that of the record GETKY failed to retrieve. If a file contains records with duplicate keys, the GETKY macro retrieves the first of these records.

GETR GET RECORD REVERSE (TYPE R)

The GETR macro retrieves the next record in the file in reverse order (i.e. toward the beginning of the file).

Operation	Operands
GETR	{fcbaddr / (1)} [, {area / (0)}] [, {LOCK / NOLOCK}]

fcbaddr — This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) — The FCB address is located in register 1.

area — This operand specifies the address of the area into which the record is to be moved (exit address).

(0) — The address of the area into which the record is to be moved is in register 0.

LOCK — The data block containing this record is to be locked once the record has been retrieved.

NOLOCK — The data block containing this record is not to be locked following execution of this macro.

Programming notes:

- The "area" operand is ignored if the IOREG operand was specified in the FCB.
- If a record outside the file is requested, the user is given control via EOFADDR (see the EXLST macro). The program can switch from GET to GETR and vice versa at any time, without the file having first to be repositioned to the end or the beginning of the file, as the case may be.
- If a GET macro supplies a record with the key K_n, and the subsequent macro is a GETR, then the next record to be retrieved is the one with the key K_{n-1}, subject, however, to the following formula:

  $K_{n-1} < K_n$

- If the file contains records with duplicate keys, the duplicates are retrieved on a "last in, first out" basis if the GETR macro is issued.
- If the file was positioned to an existing record using SETL KEY, a subsequent GETR macro retrieves this record.

INSRT INSERT RECORD (TYPE R)

The INSRT macro transfers a logical record from the user's area to the file (in the position determined by the value of its record key).

Operation	Operands
INSRT	{fcbaddr / (1)}, {area / (0)}

fcbaddr — This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) — The FCB address is located in register 1.

area — This operand specifies the address of the logical record to be inserted in the file (exit address).

(0) — The address of the record to be inserted in the file is in register 0.

Programming note:

If the file already contains a record with the specified key, the new record is not transferred (irrespective of whether DUPEKY=YES was defined in the FCB). Control is passed to the user via DUPEKY (see the EXLST macro).
The record must be retrieved in the area determined by the "area" operand even if IOREG was specified in the FCB.

ISREQ UNLOCK DATA BLOCK (TYPE 0)

The ISREQ macro is used to unlock a data block in an ISAM SHARUPD file. A data block remains locked if it has been read, but not updated and rewritten, i.e if no further action macro is issued after the read macro.

Operation	Operands
ISREQ	{fcbaddr / (r)},ACTION=UNLOCK

fcbaddr — This operand specifies the address of the FCB associated with the file in which a data block is locked.

(r) — The FCB address is located in register "r".

ACTION=UNLOCK — This operand specifies the action to be performed. Both operands must be specified to effect unlocking.

Note:

The ISREQ operand UNLOCK need not be used if a further action macro is to be issued for a SHARUPD file. A data block is then unlocked implicitly by ISAM before the action macro is executed. After unlocking by means of ISREQ, ISAM always returns to the next macro. The ID1ECB field contains one of the following error codes, which affect register 1 as follows:

ID1ECB	R1	Meaning:
0	not changed	Data block unlocked
A01	changed	Data block locked in another file
A02	not changed	No data block to unlock
AA3	not changed	FCB or address invalid

In the case of error code A01, register 1 specifies the address of the FCB associated with the file in which the data block is locked. To unlock it, the user need only issue the ISREQ macro with the UNLOCK entry, specifying "register 1" as the first operand.

Example:

In RFA mode the data block has already been unlocked when A01 is specified, leaving register 1 unchanged.

```
ISREQ FCBA,ACTION=UNLOCK
CLC   ID1ECB,=X'A01'
BNE   NAME
ISREQ  (1),ACTION=UNLOCK
```

OSTAT OBTAIN INFORMATION ON OPENED FILES (TYPE R)

The OSTAT macro provides the user with information about the users who have opened a given file. The following information is supplied:

a) the number of users who have opened the file;

b) the number of users who have opened the file as an ISAM file with all the possible OPEN/SHARUPD combinations;

c) the number of users who have opened the file with some other access method for the purpose of input or updating.

Operation	Operands
OSTAT	{fcbaddr | (1)} [, {area | (0)}]

fcbaddr This operand specifies the address of the FCB associated with the file for which the OPEN mode information is to be obtained (exit address).

(1) The FCB address is located in register 1.

area This operand specifies the area into which the information is to be moved (exit address).

(0) The address of the area into which the information is to be moved is in register 0.

Programming notes:

- Once the macro has been processed, register 1 will contain the address of the FCB and register 0 will contain the address of the area to which the information was moved. The low-order bytes of register 15 will contain zero or an error code.

- The user must have opened the file before issuing the OSTAT macro. The user's OPEN/SHARUPD combination is also included in the information.

- The area into which the information is moved is defined by the dummy section (DSECT) IDOST, a description of which can be obtained by the IDOST macro expansion.

PUT WRITE RECORD (TYPE R)

The PUT macro presents a logical record to the system to be included in the output file.

Operation	Operands
PUT	{fcbaddr / (1)} [, {area / (0)}]

fcbaddr — This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) — The FCB address is located in register 1.

area — This operand specifies the current address of the logical record to be written to the output buffer (exit address).

(0) — The address of the record to be written to the output buffer is in register 0.

Programming notes:

- The "area" operand is ignored if the IOREG operand (i.e. locate mode) was specified in the FCB.

- The PUT action macro should be used only to add records to the end of an existing file. If a file is opened in INOUT or OUTIN mode and the last macro for the file was not a PUT, PUT causes a SETL macro to be performed at the end of the file before the record is written to that file.
  Issuing a PUT macro in locate mode has the following results:

  a) The record previously generated in the buffer is checked and added to the file.

  b) The address at which the next record is to be generated is placed in IOREG.

  Since a valid IOREG value must be supplied before a record can be generated, a dummy PUT macro is issued before creating the first record in the file. Similarly, a valid record must be generated after the last PUT macro prior to closing the file. Otherwise a CLOSE error exit will be taken or a corrupted record will be appended to the file.

- In the case of format-V records, the value of the FCB field ID1RECSI must equal or exceed the size of the records added to the file.
  This can be achieved by specifying, in the RECSIZE operand of the FCB, a value equal to or greater than the maximum size of all records to be created. If RECSIZE is not defined, the value of BLKSIZE is used.

- This macro ensures that the key of any record added to the file is equal to or greater than the current highest key value. If the strict key sequence is broken, the user receives control at one of the following exits (see also the EXLST macro):

  - at the DUPEKY exit if the key of the current record is equal to the key of the preceding record and DUPEKY=YES is not specified.

  - at the SEQCHK exit, if the key of the current record is less than the highest key of the existing file.

- In the case of simultaneous updating, the last logical block in the file remains locked by a task until another action macro (apart from PUT) is called by this task.

PUTX REPLACE RECORD (TYPE R)

The PUTX macro replaces logical records of a file. The record to be replaced must first be retrieved by a GET, GETR, GETFL or GETKY macro.

Operation	Operands
PUTX	{fcbaddr / (1)} [, {area / (0)}]

fcbaddr — This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) — The FCB address is located in register 1.

area — This operand specifies the address of the logical record to be written to the output buffer (exit address).

(0) — The address of the record to be written to the output buffer is in register 0.

Notes:

- In locate mode, a PUTX macro does not cause the buffer to be rewritten. An identifier indicating that the buffer has been changed is simply added. Output does not occur until the next macro is issued, as soon as an SVC (e.g. GETKY, GET, GETR) is issued within the same FCB.

- The record key cannot be changed during the update process. If the key is changed in move mode, the user receives control at the SEQCHK exit (see the EXLST macro).
  If, during an update process in locate mode, the record key is changed or the program inadvertently modifies other portions of the data buffer, the results of subsequent processing will be unpredictable. If the data buffer is altered so that it can no longer be processed by ISAM, the user is given control at the USERERR exit (see the EXLST macro).

- The length of format-V records can be modified by the user only if he is operating in move mode. No other ISAM action macro can be issued for the file between the PUTX macro and the associated preceding GET, GETKY, GETFL or GETR macro.

- If a PUTX macro is issued in move mode for a record within a series of records having duplicate keys, the position of the record within the sequence can be altered if the modification involves increasing the original record size.

RETRY REPOSITION IN A FILE (TYPE 0)

If the PGLOCK exit of the EXLST macro is activated, the internal pointers are not correctly positioned unless the macro which led to this exit was a PUTX or ELIM (without KEY). Certain ISAM macros (GET, GETR, GETFL) will have updated the pointer before they branch to PGLOCK. The RETRY macro performs repositioning and can optionally reactivate the macro which led to the PGLOCK exit.

Operation	Operands
RETRY	FAIL=exit-addr[,ACTION={POS | RETRY | WAIT}][,COUNT=integer]

FAIL=exit-addr — This operand specifies the address which is to be given control if RETRY fails or it takes 30 minutes or more to access a block.

ACTION= — This operand specifies the action to be taken by RETRY.

=POS — The pointer is to be repositioned in the file.

=RETRY — The pointer is to be repositioned in the file and the ISAM macro repeated.

=WAIT — If the operation is repeated and the block is still not available, the ISAM macro is queued. The program is not activated until the block required for this program is available. If the task waits 30 minutes to access the data block, control is given to the exit routine defined by FAIL.

COUNT= — Decimal number between 0 and 255.
This operand specifies how often the repositioning/macro repetition is to be retried before control is transferred to the address defined by FAIL.

Default value: 1

Notes:

- This macro can be used only with the PGLOCK exit. If it is used elsewhere, a USERERR exit is taken, and ID1RTNAD in the FCB is left pointing to the last instruction in the RETRY macro expansion (unconditional branch to the FAIL address).

- If ACTION=POS is specified and positioning is successful, control is returned to the job at the statement following the RETRY macro.
  If ACTION=RETRY is specified and the positioning/macro repetition is successful, control is returned to the job at the statement following the ISAM macro which was reissued.
  If either action fails (even after being retried the number of times specified by COUNT), control is passed to the job at the location specified by the FAIL operand.

- With the PGLOCK exit the job may use any register except register 1. If control is returned to the job as a result of RETRY, the job's registers have the same contents as they would have had if the action macro had been successful immediately.

- If the last record successfully accessed prior to the PGLOCK exit is one of a series of records with the same key, repositioning will position the task to the first record in this series.

SETL POSITION FILE (TYPE R)

The SETL macro enables positioning to take place to the beginning or end of a file or to any record determined by a specified key.

Operation	Operands
SETL	$\left\{\begin{matrix}\text{fcbaddr}\\ \\ (1)\end{matrix}\right\}, \left\{\begin{matrix}\left\{\begin{matrix}B\\E\\KEY\end{matrix}\right\}\\ \\ (0)\end{matrix}\right\}$

fcbaddr — This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) — The FCB address is located in register 1.

B — Positioning is to take place to the beginning of the file.

E — Positioning is to take place to the end of the file.

KEY — Positioning is to take place to the key defined by the KEYARG operand of the FCB.

(0) — Prior to the execution of SETL, the position code must be placed in register 0 as follows:

B 0

E 1

KEY Address of the KEYARG field in the FCB

Default value: B

Programming notes:

- If the KEY option is used and the specified key does not exist, positioning to the record with the next higher or lower key still occurs, without resulting in an error; i.e. the next GET macro reads the next higher record, and the next GETR macro reads the next lower record.
- If the KEY option is used and the specified key exists, both the GET macro and the GETR macro read the associated record.
- If records with duplicate keys exist in a file, the position indicators are set so that the next GET macro fetches the record which was the first duplicate record written to that file.
- Performing SETL B for a null file causes control to be transferred to the EOFADDR error exit of the EXLST macro.

STORE STORE RECORD ‹TYPE R›

The STORE macro obtains a logical record from the user area and places it in the file at the position defined by the key.

Operation	Operands
STORE	{fcbaddr / (1)}, {area / (0)}

fcbaddr — This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) — The FCB address is located in register 1.

area — This operand specifies the address of the logical record to be stored in the file (exit address).

(0) — The address of the record to be stored in the file is in register 0.

Programming notes:

- STORE has the same function as the INSRT macro. If, however, a record with the same record key is found in the file, the new record overwrites the existing one. No DUPEKY exit is taken.
- If DUPEKY=YES is specified in the FCB, duplicate keys are allowed. The new record is then placed in the file immediately after any other record with the same key.
- The record to be stored must be retrieved in "area" even if the file is processed in locate mode (IOREG).

6.4 SAM (SEQUENTIAL ACCESS METHOD)

SAM provides a means of accessing records sequentially, beginning at a specified point. This organization is most useful for making a search through all the records of a file, including retrieving and updating a specific record and returning it to the file. Records can also be added to the file. Unlike ISAM, no additional space is required other than that needed by the logical records themselves.

SAM automatically performs all blocking, deblocking and buffering for the user. If the user requests the system to utilize only one I/O area, then no buffering (overlapping) takes place.

Logical records are retrieved by means of the GET macro. SAM expects records to be required in the same sequential order in which they were written. Logical records designated for output are output by means of the PUT macro. The program can continue as if writing of the data record would follow immediately, although the access method's routines may perform blocking with other logical records and delay the actual writing until the output buffer has been filled. Buffers are automatically maintained by the system. SAM is largely device-independent and allows files to be processed on disks and tapes.

Transfer of data to and from the file can be effected in either of two modes:

Move mode: The user specifies the address of the record in his program, and the system is responsible for transferring it to or from the buffers.

Locate mode: The user requests the address of the current record in the buffer area via a register.
The user is responsible for transferring data to or from the buffers. In locate mode, direct access to a record is possible via the retrieval address in the FCB (see section 6.4.3).

The following action macros are available in SAM to control file processing:

GET Retrieves the next record from the file in physically sequential order.
PUT Writes a logical record to the file.
PUTX Returns an updated logical record to the file (direct access only).
RELSE Causes any remaining logical records in a buffer to be bypassed for input. On output, the next logical record created is written as the first record of a new buffer.
SETL Specifies the position at which subsequent file processing is to start.

For files opened in OUTPUT or EXTEND mode, SAM interprets each PUT or SETL macro as an end-of-file indicator. The last PUT or SETL macro prior to CLOSE for a file thus automatically defines an end-of-file indicator for the system. If the user wishes to delete all of the records beyond a given record, he can use the SETL macro to position to the desired point in the file, and then issue a CLOSE macro to close that file.

For direct access, a retrieval address is made available to the user. The format of this retrieval address is described in detail in section 6.4.3, "FCB Retrieval Address". When a logical record is stored (by means of a PUT macro), its retrieval address is made available in the FCB. The user can, if he wishes, create another file from this retrieval address data and thereby establish a basis for subsequent non-sequential processing of the file being created. This retrieval address is also available in the FCB after the execution of a GET macro. Thus, even if the user did not create the file, he can still create a secondary file from the retrieval addresses to facilitate subsequent non-sequential processing of the original file.

6.4.1 SAM Record Formats

SAM supports three record formats, as follows:

F (for fixed-length records)
V (for variable-length records)
U (for records of undefined length)

See section 6.1.3 for the format of logical records.

Note:

For format-U records, SAM uses one logical record per physical block (buffer). For example, a user who specifies standard blocks (2048 bytes) and outputs 48-byte logical records would waste 2000 bytes.

The size of the logical record must not exceed the buffer size (see the BLKSIZE operand).

6.4.2 Opening a SAM File

The file can be opened in the following modes:

INPUT — Retrieve records from an existing file in the direction of the end of the file.

OUTPUT — Create a new file or replace an existing file.

EXTEND — Add records to the end of an existing file.

UPDATE — Retrieve and replace records in an existing file (locate mode only).
The PUTX macro is used to rewrite logical records. Each record, however, must first be retrieved by a GET macro in locate mode. The user must not change the length of these records.

REVERSE — Retrieve records from an existing file in a reverse direction. Multivolume files cannot be opened in REVERSE mode.

OPEN mode	INPUT	OUTPUT	EXTEND	UPDATE	REVERSE
GET	X			X	X
PUT		X	X		
PUTX				X	
RELSE	X	X	X	X	X
SETL	X	X	X	X	X

Table 6-8: SAM action macros in relation to OPEN mode

Programming notes:

- If a SAM file is created or replaced (OPEN=OUTPUT), both the primary and the secondary allocation must be a multiple of the block size.
- In the case of OUTPUT and move mode operation with
  - variable-length records (format V) or
  - fixed-length records (format F) and more than 1 record per block

  the primary allocation must be at least twice the block size. If these requirements are not met, control passes to the EXLST error exit NOSPACE (insufficient memory space).
- With OUTPUT and locate mode, the user receives in IOREG the location at which the first record starts, as soon as OPEN has been performed. If the first PUT macro has been performed, the user receives from IOREG the start of the second record etc.

6.4.3 FCB Retrieval Address

The P1FCB field ID1RPTR (one word) contains the retrieval address in the form bbbbbbrr, where:

bbbbbb Number of the buffer in the file.

rr Number of the logical record within the buffer. The first record in the file has a retrieval address of 00000101.

OPEN mode	Initial value	SETL B	SETL E
INPUT, UPDATE	unchanged	unchanged	bbbbbb01
OUTPUT	00000100	00000100	error
EXTEND	bbbbbb00	00000100	error
REVERSE	unchanged	unchanged	unchanged

Table 6-9: Retrieval address values after OPEN, SETL B and SETL E

"bbbbbb" stands for:

- with EXTEND: the highest buffer number +1 in the file.

- with INPUT/UPDATE: the highest buffer number in the file.

"unchanged" means that ID1RPTR is not updated until after the GET macro following the SETL macro.

The other action macros have the following effect as regards the retrieval address:

GET If the specified record makes it necessary to retrieve a new buffer, bbbbbb identifies the buffer number for the record and rr is reset to 00.

PUT If the specified record makes it necessary to write the buffer, bbbbbb identifies the buffer number for this record and rr is reset to 00. In other words, the value is set to the number of the buffer in which the record is to be placed.

RELSE If the file is opened in OUTPUT or EXTEND mode, bbbbbb is set to the number of the buffer in which the next record will be placed; rr is set to 00.

Notes:

- This field is maintained only for tape files created with standard blocks.

- bbbbbb is buffer-oriented.

  Given:

  BLKSIZE=(STD,2)
  RECFORM=F
  RECSIZE=512

  The retrieval address for the 10th record is 00000202. The retrieval address for the 20th record is 00000304.

- The system never increments the rr field. As described above, it is reset to zero when a new buffer is to be processed. The user can increment the rr field if he wishes to maintain retrieval address information for each logical record (IDFCB field ID1RPTR).

6.4.4 Use of PAM Keys in SAM

The format of the rightmost eight bytes of the PAM key in SAM files is BBBXLLXX.

where:

BBB	Buffer number in binary
X	Not used
LL	Length of useful data in the buffer

The first X-byte (byte 4) is reserved for future system use.

Note:

A modification will be made in respect to the support of preformatted disks (FBA disks), for which reason a user program cannot entirely count on obtaining information in exactly the form described above.

6.4.5 SAM Macros

FCB CREATE FILE CONTROL BLOCK (TYPE 0)

The following operands may be supplied in the FCB for SAM. A detailed description of these operands can be found in section 5.4, "Service Macros for Disk Files".

Operation	Operands
FCB	FCBTYPE=SAM [,BLKSIZE={STD | (STD,absexp)}] [,EXIT={(relexp) | relexp}] [,FILE=filename] [,FORM=SHORT] [,IOAREA2={NO | relexp}] [,IOREG=regno] [,LINK=linkname] [,OPEN={INPUT | OUTPUT | EXTEND | REVERSE | UPDATE}] [,PASS={password | absexp}] [,RECFORM={{F | V | U} | ({F | V | U}[,{A | M | N}])}] [,RECSIZE=absexp] [,RETPD=days] [,VARBLD=regno] [,WRCHK={YES | NO}] [,OPTION={code | (code,code)}]

See chapter 5 for a description of the operands.

GET GET RECORD (TYPE R)

The GET macro retrieves the next sequential record of a file.

Operation	Operands
GET	$\left\{\begin{matrix}\text{fcbaddr}\\(1)\end{matrix}\right\}[,\left\{\begin{matrix}\text{area}\\(0)\end{matrix}\right\}]$

fcbaddr — This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) — The FCB address is located in register 1.

area — This operand specifies the address of the area into which the record is to be moved if processing is in move mode (exit address).

(0) — The address of the area into which the record is to be moved is located in register 0.

Programming notes:

- The "area" operand is ignored if the IOREG operand was specified in the FCB.
- If a record beyond the end of the file is requested, the user is given control at EOFADDR (see the EXLST macro).
- The "area" operand is ignored for files opened in UPDATE mode.
- In the case of SAM files, the access time can be minimized by using large buffer areas. SAM itself then chains together consecutive PAM blocks (chained I/O).
  However, worthwhile savings can be expected only in the case of files which have a substantial proportion of contiguous storage space (see the SPACE operand in the FILE command).

Note:

A modification will be made in respect to the support of preformatted disks (FBA disks), for which reason a user program cannot entirely count on obtaining information in exactly the form described above.

PUT WRITE RECORD (TYPE R)

The PUT macro presents a logical record to the system for inclusion in an output file.

Operation	Operands
PUT	$\left\{\begin{matrix}\text{fcbaddr}\\(1)\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{area}\\(0)\end{matrix}\right\}\right]$

fcbaddr — This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) — The FCB address is located in register 1.

area — This operand specifies the address of the logical record to be written to the output buffer (exit address).

(0) — The address of the record to be written to the output buffer is in register 0.

Programming notes:

- The "area" operand is ignored if the IOREG operand was specified in the FCB.
- If this macro is issued in locate mode, the system places the address of the first available area within the output buffer in the register specified in IOREG. The user must then ensure that the record to be included in the output file is made available at that address.
- If format-V records are processed in locate mode, the system places, in the register specified by the VARBLD operand, the buffer capacity remaining after every PUT. It is the user program's responsibility to ensure that the next record can be completely accommodated in the remaining buffer area. If it cannot, a RELSE macro must be issued.

Note:

A modification will be made in respect to the support of preformatted disks (FBA disks), for which reason a user program cannot entirely count on obtaining information in exactly the form described above.

PUTX REPLACE RECORD (TYPE R)

The PUTX macro is used to return an updated logical record to a file. The user must not change the length of the record during the replacement process.

Operation	Operands
PUTX	{fcbaddr / (1)}

fcbaddr This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) The FCB address is located in register 1.

Programming notes:

- The PUTX macro can return only a record previously supplied by a GET macro in locate mode. The file must be opened in UPDATE mode.
- A macro does not lead to an SVC in every instance. An SVC is only issued by the buffer management routine when a buffer is to be output. If the macro STIXIT SVC=... or STIXIT SYSLIST=... is issued, the user cannot expect to be given control in the STIXIT task each time he issues PUTX.

RELSE CLOSE BLOCK (TYPE R)

The RELSE macro causes

- (with input files) the remaining records in the buffer to be skipped.
- (with output files) the buffer that has just been processed to be closed and the next record to be written to a new buffer.

Operation	Operands
RELSE	{fcbaddr / (1)}

fcbaddr This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) The FCB address is located in register 1.

Programming notes:

- If the file is opened in INPUT, REVERSE or UPDATE mode, the RELSE macro causes any records remaining in the buffer to be bypassed when the next GET macro is issued.
- If the file is opened in OUTPUT or EXTEND mode, the RELSE macro causes the next record to be written as the first record of a new buffer when the next PUT macro is issued.
- If the file is opened in UPDATE mode and a PUTX macro is issued, RELSE causes the entire buffer to be rewritten.
- If the last action macro issued was RELSE, the RELSE macro is ignored. This condition overrules conditions 1 and 2 above.

SETL POSITION FILE (TYPE R)

The SETL macro is used to specifiy the position of the internal record indicator from which subsequent file processing is to start.

Operation	Operands
SETL	{fcbaddr / (1)} [, {B / E / R}]

fcbaddr — This operand specifies the address of the FCB associated with the file to be processed (exit address).

(1) — The FCB address is located in register 1.

B — Positioning is to take place to the beginning of the file.

E — Positioning is to take place to the end of the file.

R — The positioning information is to be obtained from the retrieval field in the FCB.

Programming note:

An illegal SETL operand causes control to be passed to the USERERR address of the EXLST macro.

6.5 EAM (EVANESCENT ACCESS METHOD)

EAM is the access method for temporary files in BS2000.

EAM has the following characteristics:

- EAM files are not cataloged. Consequently, opening an EAM file does not involve disk access.
- An EAM file is automatically erased after completion of the job which opened it (temporary file).
- Communication between EAM and user can take place only via the EAM control block (MFCB, or Mini-FCB). Modification of the MFCB at file opening time is not catered for.
- No label processing or storage space allocation takes place.
- EAM uses only public volumes.
- Space requirements for EAM routines and delays occurring during read/write access operations are less than with the standard access methods for cataloged files (UPAM, SAM, ISAM).
- An EAM file can be processed by one job only (no shared file update). A single job can, however, open and process several EAM files.
- EAM transfers blocks which are 2048 bytes in length. In the case of chained I/O, up to 16 consecutive blocks can be processed with one macro.
- EAM files are not supported by the checkpoint/restart routines of the system.

6.5.1 EAM Functions

EAM can perform the following operations:

- Set up and open a new file.
- Reopen an existing file.
- Read (sequential or direct).
- Write (sequential or direct).
- Check for termination of an I/O operation.
- Check and wait for termination of an I/O operation.
- Close a file.
- Erase a file.

A particular operation is selected by specifying a hexadecimal code (operation code) in the MFCB. The EAM macro serves to initiate the specified operation. The effect is determined by the MFCB fields which EAM additionally evaluates after it has identified the operation code.
Table 6-10 summarizes the EAM functions.

MFCB fields / Operation	Operation code (hex.)	Option byte	Logical block number	File name	Sense byte	Status byte	Address of I/O area	Number of blocks to be transferred	Address of I/O area 2
Create and open new file	00	E	i +	S	S	i	E	E-	E
Reopen existing file	01	E	S	E+	S	i	E	E-	E
Read	02	E	E	E	S	S	i	S	i
Write	03	E	E	E	S	S	i	i	i
Check	04	i	i	E	S	S	i	i	i
Check and wait	08	i	i	E	S	S	i	i	i
Close file	06	i	S	E	S	S	i	i	i
Erase file	05	i	i	E	S	i	i	i	i

Table 6-10: EAM functions

E: Contents of field are evaluated
S: Contents of field are set
i: Contents of field are ignored
+: For exceptions affecting object module files, refer to section 6.5.3, "Programming Notes"
-: For exceptions, see description of field

6.5.2 EAM Macro (Type R)

There is only one supervisor call (SVC) with which to invoke EAM. The operation to be performed is defined by the contents of the MFCB.

Operation	Operands
EAM	{mfcbaddr (1)}

mfcbaddr This operand specifies the address of the MFCB.

(1) The address of the MFCB is located in register 1.

Description of the MFCB fields

The MFCB is the communication area between EAM and the user. The field organization is shown in Table 6-11.

The MFCB must be aligned on a word boundary.

Field description	Symbolic field name
Operation	IDMFOPC
Option byte	IDMFOC
Logical block number	IDMFLBN
File name	IDMFFN
Sense byte	IDMFEB
Status bytes	IDMFSB
Reserved for system	
Address of I/O area 1	IDMFIO1
Number of blocks to be transferred	IDMFNHP
Address of I/O area 2	IDMFIO2

Table 6-11: MFCB field organization

The MFCB can be provided with symbolic names by means of the IDMCB macro.

- **Operation codes**

  - Create and open a new file - DMFO

    EAM assigns a file name (binary number in the range 1 - 14000), creates several entries in its own tables, and checks the option byte, the size of the I/O area and the MFCB address.
    Exceptions affecting object module files are described in section 6.5.3, "Programming Notes".

  - Reopen an existing file - DMFRO

    The option byte and the size of the I/O areas are rechecked.
    In this case, the logical block number can be set by EAM to either beginning or end of file, depending on the value of bit 2^1 in the option byte.

  - Read - DMFRD

    Chained input:

    This read mode is specified by setting bit 2^2 in the option byte.
    The number of blocks read will be as specified in the field "Number of blocks to be transferred".
    The field "Logical block number" specifies the first block to be read; a specification of 0 causes blocks to be read which immediately follow the last block referenced (sequential processing).
    In the case of chained input, it may be the case that one or more referenced blocks are located beyond the end of the file.
    In this case the remaining blocks belonging to the file are transferred, the number of blocks involved being entered in the field "Number of blocks to be transferred", and either bit 2^2 or 2^6 being set in the error byte. The number of blocks transferred can be 0.

    Non-chained input:

    This read mode is specified when bit 2^2 in the option byte is not set.
    Either the block with the specified block number is read, or the block following the last block referenced is read (logical block number = 0; sequential processing).

  - Write - DMFWR

    Chained output:

    This output mode is specified by setting bit 2^2 in the option byte.
    The number of blocks written will be as specified in the field "Number of blocks to be transferred".
    The field "Logical block number" specifies the position in the file to which the first of a series of blocks is to be written; specifying 0 in this field causes blocks to be written directly following the end of the file.

    Non-chained output:

    This output mode is specified when bit 2^2 in the option byte is not set.
    A block is written to the position specified in the field "Logical block number"; specifying 0 in this field causes a block to be written directly following the end of the file.

- Check - DMFCK

  A check is performed as to whether the last I/O operation has been completed.
  If the last I/O operation has terminated, the status bytes are transferred to the MFCB.
  If the last I/O operation has not yet been completed, register 15 is set to X'00000008'.
  Whatever the case, control is immediately returned to the user.

- Check and wait - DMFCW

  Completion of the last I/O operation is awaited, and the status bytes are subsequently transferred to the MFCB.
  If the transfer of the status bytes has already been implemented by another operation, the current operation has no effect.

- Close file - DMFCL

  The file is closed but not erased.
  The block number of the last block in the file is placed in the field "Logical block number".

- Erase file - DMFER

  The file is erased whether it is open or not.

- **Option bytes**

  0
  2 This bit is evaluated when reading and writing.

  =0 I/O area 1 is to be used.
  =1 I/O area 2 is to be used.

  1
  2 This bit is evaluated when reopening a file.

  =0 The logical block number of the last block belonging to the file is placed in the field "Logical block number".
  =1 The value zero is placed in the field "Logical block number".

  2
  2 This bit is evaluated when opening and reopening a file.

  =0 Chained I/O is not to be used.
  =1 Chained I/O is to be used.

  3
  2 Reserved

  4
  2 This bit is evaluated when opening and reopening a file.

  =0 The file to be opened is not the object module file of the calling job.
  =1 The object module file of the calling job is opened.

  5 7
  2 - 2 Reserved for use by the system.

- **Logical block number**

  The logical block number is a 2-byte binary number which indicates the block within the file which is to be read or written.

  In the case of chained I/O, the binary number denotes the first in a series of blocks to be read or written.

  The value of this binary number can be in the range 1 - 65535.

  The value 0 specifies sequential processing:

  - In the case of reading, the block which directly follows the last block referenced (in a read or write operation) is transferred.
  - In the case of writing, a block is added to the end of the file.
  - In the case of chained I/O the same applies by analogy to a series of blocks instead of to a single block (see also operation code: Read - DMFRD).

  Each block contains 2048 bytes. Any record structure within a block remains unknown to EAM (same as with UPAM).

- **File name**

  The file name is a 2-byte binary number which is placed in this field by EAM when a new file is opened. This number must be specified whenever the file is subsequently referenced.

  The value of the binary number is restricted to the range 1 - 14000.

- **Sense byte**

  If an operation could not be successfully executed, this field will contain an indication of the cause; in this case, EAM sets register 15 to X'00000004'. After normal execution of an operation, register 15 contains X'00000000'; the sense byte is set to X'00'.

  The bit positions of the sense byte have the following meanings:

  0_2 Illegal operation:

  - Illegal operation code
  - An existing file is not open and an attempt was made to close the file or to perform a read/write operation.
  - MFCB is not aligned on a word boundary.
  - Bit 2_2 in the option byte is set (chained I/O) and the value in byte DMFNHP (number of blocks to be transferred) is not in the range 1-16.

  1_2 Illegal file name:

  - The file name is not in the range 1 - 14000.
  - The file does not exist.
  - The file does not belong to the calling job.

2^{2} Illegal block number:

- A read operation has been requested which refers explicitly (i.e. block number ≠ 0) to one or more blocks located beyond the end of the file.

- A write operation has been requested with a block number greater than:
  (number of last block in file) + 1.

2^{3} Illegal I/O area:

- Illegal address of an I/O area.

- One of the I/O areas is too small.

2^{4} EAM space is not available.

- The number of EAM files in existence has already reached the maximum of 14000 (upon opening a file).

- The number of occupied disks of all EAM files of the system has reached the maximum of 193,536.

2^{5} A non-privileged program is attempting to process a privileged file.

2^{6} A read operation has been requested which refers implicitly (i.e. block number = 0) to one or more blocks beyond the end of the file.

2^{7} The last I/O operation requested, or the preceding one, was unsuccessful. The status bytes should be checked to determine the cause of the error which has occurred.

- Error on starting the last I/O operation requested. In this case, all 5 status bytes contain the value X'00'; a check operation performed immediately afterwards has no effect. In this case, however, the preceding I/O operation has been successfully completed (otherwise the last I/O operation requested would not even have been started).

- In all other cases (i.e. at least one status byte ≠ 0), an error occurred in the preceding I/O operation; the status bytes then contain additional information concerning the error.

  The last I/O operation requested was not started.

- **Status field**

This field is set only if the following conditions are simultaneously satisfied:

- The preceding operation was a read or write operation.

- The current operation is a read, write, check, check-and-wait, or close operation.

The following bytes are transferred from the Channel Control Block (CCB):

- Standard device byte
- 3 sense bytes
- Executive flag byte (cf. Appendix A.3)

- **Address of I/O area 1**

  This field contains the virtual address of the first byte of I/O area 1.

  In write operations, a block is transferred from this area to the file; in read operations, the process is reversed.
  With non-chained I/O, the area must be located in a virtual page and be aligned on a word boundary.

  In the case of chained I/O, the area must be aligned on a page boundary and must be able to accommodate at least as many blocks as are to be transferred.

- **Number of blocks to be transferred**

  This is a 1-byte binary number in the range 1-16. This field is evaluated upon opening or reopening a file only if chained I/O is specified in the option byte.

  In the case of chained I/O and upon reaching end of file in read mode, the number of blocks transferred is contained in this field (see also operation code: DMFRD).

- **Address of I/O area 2**

  This field contains the virtual address of the first byte of I/O area 2. This address may be the same as that for I/O area 1. However, if chronologically overlapping processing is to be performed, this field must contain the address of an area which does not encroach onto I/O area 1. The conditions affecting I/O area 1 also apply here.

6.5.3 Programming Notes

- Use of check operations

  After a read or write call, control is returned to the user as soon as the requested operation has been accepted. In other words, this operation does not need to have been completed.

  However, before a read or write operation is accepted, completion of the previous read/write operation (if any) is awaited (with an implicit check-and-wait operation).

  Similarly, completion of the last read/write operation is awaited if a close operation is requested.

  Thus, a check operation is necessary only after the last in a series of read/write operations if the file is not immediately closed again or if, in the case of chained I/O, reading is continued until the end-of-file condition is reached (bit 2^2 or 2^6 in the sense byte = 1) and the number of blocks transferred is not equal to 0.

Example:

- 3 read operations are performed in an EAM file. The file is not subsequently closed, as it is still required for later input/output operations; however, these I/O operations are not requested until the blocks which were read have been processed:

```
READ
READ
READ

CHECK/WAIT                    Termination of the previous I/O operation
  .                           is awaited.
  .
  .
Processing of the blocks read
  .
  .
  .
Further I/O operations
```

- A read operation with chained I/O is performed from an EAM file (6 blocks at a time). Bit 2^2 is set in the sense byte.

  The field "Number of blocks to be transferred" now contains, for instance, the value X'02', i.e. of the 6 blocks requested, only the first two belong to the file, and only those two are transferred.

  In this case, one of the following operations must be requested:

  A check-and-wait operation, or
  a close operation.

  In both cases, termination of the read operation is awaited and the status bytes are placed in the MFCB.

- Changing the input/output mode (i.e. chained or non-chained I/O), the addresses of the I/O areas and the number of blocks to be transferred.

  Upon opening or reopening a file, the following information is saved in system memory:

  - Addresses of the I/O areas
  - Number of blocks to be transferred
  - I/O mode

  If any of this information is to be changed during processing of the file, the following sequence of operations is required:

  1. Close file.
  2. Modify fields in MFCB.
  3. Reopen file.

  The number of blocks to be transferred should be altered, for example, if fewer blocks are to be transferred with the last write operation than was specified when opening the file.

Example:

99 blocks are to be written to an EAM file; 15 is selected as the number of blocks to be written by one EAM call (byte DMFNHP in the MFCB).

The following operations are then requested:

OPEN (new file)

WRITE
WRITE
WRITE
WRITE
WRITE
WRITE

After these operations have been performed, only 9 blocks remain to be written. Therefore, the file must be closed and reopened, with the value 9 for the number of blocks to be transferred.

CLOSE

REOPEN

WRITE

CLOSE

- Overlapped input/output

A reduction in processing time can be achieved by means of the parallel execution of user job and I/O operations.

While a portion of the data in a file is being processed, a further I/O operation can be executed. Two I/O areas must be employed for this to be possible.

The following example illustrates the overlapping of processing and input operations.

EAM

Example:

```
START
l:=1,END-OF-FILE:=NO
READ BLOCK l into I/O area 1
End-of-file?
Y                                                    N
        LOOP
            l:=l+1
            READ block l into I/O area 2
            End-of-file?
            Y                                        N
            END-OF-FILE:=YES
            PROCESS block l-1
            IF END-OF-FILE=YES, TERMINATE
            l:=l+1
            READ block l into I/O area 1
            End-of-file?
            Y                                        N
            END-OF-FILE:=YES
            PROCESS block l-1
            IF END-OF-FILE=YES, TERMINATE
END
```

Fig. 6-9: Overlapped input/output

- Sequential read with chained input/output

  Chained I/O, when used in conjunction with sequential reading, is most efficient if:

  - a multiple of 3 is selected as the number of blocks to be transferred (e.g. 3, 6, 9, ...,3*n,...);
  - the block number selected is in the form (multiple of 3) + 1 (e.g. 1, 4, 7, ..., 3*n+1,...).

- Handling object module files using EAM

  Each job can process exactly one object module file. If bit 2^4 in the option byte is set, all operations relate to the object module file.

  The actions involved in opening or reopening a file are illustrated in the following diagram:

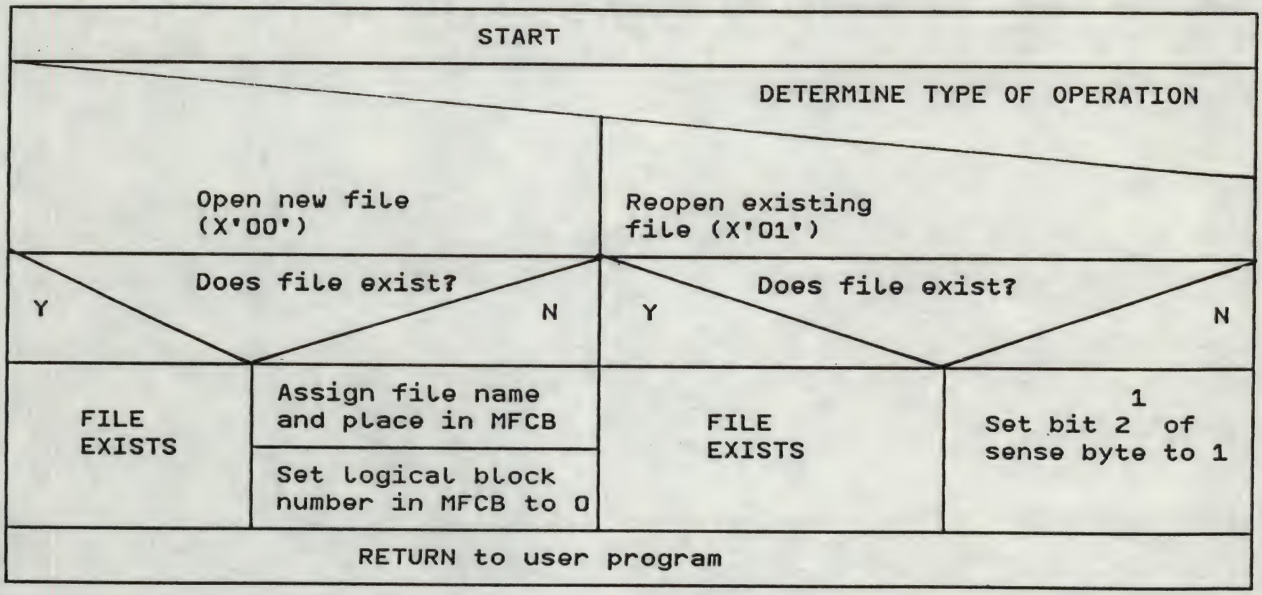


Fig. 6-10: Actions involved in file opening

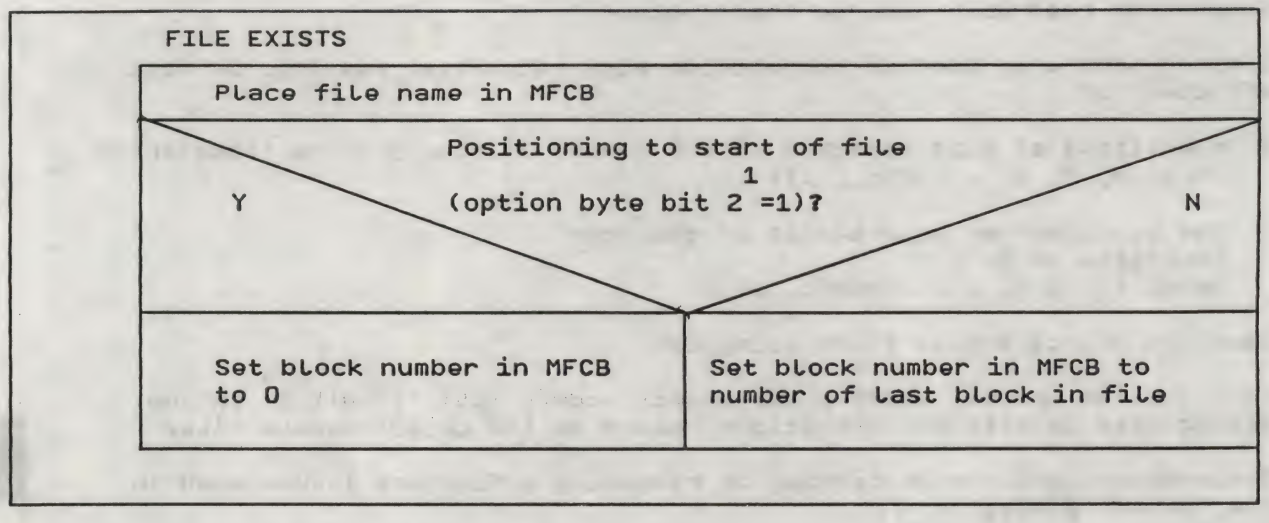


Fig. 6-11: Procedure for opening the object module file

A APPENDICES

A.1 COMMAND AND MACRO CONVENTIONS

Commands

All BS2000 commands are similar in structure. They all begin with a slash (/) which identifies them for both system and user as commands. In interactive mode, the system provides the slash; in batch mode, it is entered by the user in the first column of each line or card belonging to the command. The slash is followed by various entries, which may be divided into four fields:

<——— Command ———>

/	Name	Operation	Operands	Remarks

The IDA commands constitute an exception to this, as several such commands may be combined to form an IDA statement:

<——— IDA statement ———>

/	Statement name	Operation	Operands	Remarks	;	Operation	Operands	Remarks	;

<——— IDA command ———> <——— IDA command ———>

Name field

Purpose: The command or statement name specifies the symbolic address which may be used as a branch destination in other commands. Additionally, a statement name in an AT command (IDA) identifies the IDA counter.

The user can decide whether or not to give the command a name. (For this reason, this name is not usually referred to in the descriptions of the commands, although a name can be assigned to each command.)

Conventions

Structure: The name field may commence with any number of blank characters (X'40'). A period marks the start of the name proper. This period is followed by an alphabetic character, and this, in turn, may be followed by up to 7 alphanumeric characters. A blank may appear at the end.

Examples:

```
/.NAME1......
/......XY......
/.A1234567......
```

Operation field

Purpose: The operation to be performed is specified by the command name or its abbreviation. This field must therefore be included in all commands.

Structure: The operation field can commence with any number of blanks (X'40'). If there is a name field, it must be separated from the operation field by a blank. If no name field is included, the command name or its abbreviation can immediately follow the slash.

Examples:

```
/STATUS
/.NAME____FSTATUS
/.X_STEP
```

Operand field

Purpose: The operands of a command define the various functions of that command.

Structure: A distinction is drawn between positional operands and keyword operands:

- Positional operands are defined by their position in the command and must therefore be entered in the prescribed order.
- Keyword operands are defined by a keyword (KEYWORD=) and may thus appear in any order.

The first operand in this field must be separated from the command name in the operation field by at least one blank X('40'). The commas used to separate individual operands can be preceded or followed by any number of blanks, as can the equals signs within keyword operands. This applies also to the separating characters , ; : () = + - * / < > in IDA commands.

Examples:

```
/PRINT DATA                                  positional operands
/.NEUMANN   CATALOG  NAME1  ,NAME2

/FILE_LINK=CHAIN,DEVICE=TAPE                 keyword operands
/.COMPILE   PARAM   ERRFIL=YES

/ERASE   FILE      ,DATA,LIST=YES            2 positional operands,
                                             1 keyword operand

/DISPLAY  %3  :%7                            2 positional operands
                                             in IDA commands
```

Comments field

Purpose: Any further information which the user may wish to include regarding the command may be entered in the comments field, as this field is not interpreted by the system. Such comments are optional.

Structure: The comments field may commence with blanks (X'40'). The actual comments must be enclosed within double quotation marks ("), and may include all characters except the quotes symbol itself and the ETX character. Such comments can be appended to the operand field, or inserted either before or after the comma separating operands.

Examples:

```
Incorrect use:
/FSTAT ***"FILE STATUS FOR ALL FILES"***"
Correct use:
/.PURPOSE EXECUTE  PROG  "TEST PROGRAM NO.5"
```

Continuation Lines

A command can consist of one or more lines. In the case of punch cards, each card corresponds to a line.

The hyphen serves as a continuation character, i.e. it indicates that there is a continuation line. The number of continuation lines allowed depends on the continuation mechanism (system parameter SSMCOPT at system generation time). See section 1.2 in the "Control System Command Language" manual.

The first operand of a command must be entered in the first line.

Interactive mode: If the command is entered at a terminal, the continuation character can appear anywhere within a line and must be immediately followed by the ETX symbol. The system responds with the slash for the next continuation line.

Batch mode: In batch mode, the continuation character must appear in column 72 of the line or card. The remainder of the record (columns 73 through 80) is then ignored. The user himself must enter the slash in column 1 of each line or card.

Procedures (interactive and batch mode):

Continuation lines in procedure files are subject to the same rules as for batch mode.

Examples:

Interactive mode:

```
Line 1:  /FILE  FILENAME,-<
Line 2:  /LINK=DSET10,"LINKNAME"-<
Line 3:  /DEVICE=TAPE,FCBTYPE=SAM <
```

"<" represents the ETX character.

Batch mode:

```
Line 1:  /PROCEDURE C,(&A,&COMPIL=,                    -
Line 2:  /&PASSW=C'AD',PROG,&AB1)                      -
Line 3:  /,SUBDTA=*"SYMBOL FOR PROGRAM" <
```

The hyphen "-" must be given in column 72 in lines 1 and 2.

Macros

Macro instructions are processed by the Assembler using macro definitions supplied by Siemens. Macros are written according to the conventions described below.

The processing of a macro instruction by the Assembler is called the expansion of the macro instruction. The expansion results in data fields and executable instructions. The data fields, called operands, specify the operation to be performed and are contained in either registers (operand registers) or data areas (operand lists).

- If the operands are contained in registers, only registers 0 and 1 may be used.

- If the operands are contained in an operand list, the address of this list is placed in register 1 and referenced by the service routine called.

The macro instruction language

Macro instructions, like Assembler instructions, are written in the following format:

Name	Operation	Operands
NAME	MACRO	OP1,OP2,...

Name field

The name field of the macro instruction may contain a symbol or be left blank. Normally, this symbol is the name associated with the first executable instruction of the macro expansion.

Operation field

The operation field contains the mnemonic operation code of the macro instruction. This code may be a string of not more than eight alphanumeric characters, the first of which must be a letter. User macro names should not begin with a dollar sign ("$"), since privileged system macros begin with this character.

Operand field

The operand field may contain no operands, one operand, or several operands separated by commas. There are two types of operand: positional and keyword.

Conventions

Positional operands

When three or fewer operands are required by a macro instruction, positional operands are generally used. Positional operands must be written in a specific order, e.g.:

```
EXAMPLE  A,B,C
```

At assembly time, the order in which operands A, B and C are processed is determined by the order in which they appear. If the second operand (in this case: B) is omitted, the user must supply the second comma so as to maintain the position of the third operand. The positional field should contain no blanks (X'40'), e.g.:

```
EXAMPLE  A,,C
```

If the last operand is to be omitted, the delimiting comma need not be written. For example, if the operands B and C are to be omitted, the macro instruction could be written as follows:

```
EXAMPLE  A
```

Keyword operands

The keyword associated with a given keyword operand uniquely identifies that operand to the Assembler. Therefore, these operands can be written in any order. A keyword operand is written as a keyword, as defined in each macro instruction description, immediately followed by an equals sign and its value, e.g.:

```
EXAMPLE AREA=X,LENGTH=100
```

Mixed operands

An operand field may contain both positional and keyword operands; however, all positional operands must precede all keyword operands, e.g.:

```
EXAMPLE A,B,C,AREA=X,LENGTH=100
```

The rules governing the omission of positional and keyword operands apply also to mixed operand fields. For example, if the operands B, C, and AREA are to be omitted from the above example, it must be written as follows:

```
EXAMPLE  A,LENGTH=100
```

Operand sublists

A sublist consists of one or more positional operands, each separated by commas and the total list enclosed in parentheses.
The entire sublist is considered to be one operand in the sense that it occupies a single position in the operand field or is associated with a single keyword. The contents of the sublist are processed similarly to positional operands. The following operands are sublists:

```
(A,B,C)
(A)
```

In the second example, the sublist consists of only one operand. If a macro instruction description shows that an operand is to be written as a sublist, the enclosing parentheses must be written, even if there is only one element in the sublist.

Macro description notational symbols

Notational symbols in the operand fields of macro instruction descriptions assist the user in showing how, when, and where an operand is to be written:

The notational symbols are:

- Braces {}
- Square brackets []
- Underscore
- Ellipsis, shown as...

1. Braces denote operand grouping. They are used primarily to group alternative operands, e.g.:

   {INPUT / OUTPUT}

2. Square brackets denote options. Information enclosed in square brackets may either be omitted or written in the macro instruction, depending on the requirements.

   Example:

Name	Operation	Operands
[symbol]	EXAMP	[{INPUT / OUTPUT}]

   The user has 3 options for the operand entry, i.e. to specify:

   - no operand
   - INPUT
   - OUTPUT

3. An underscore indicates a default value that is assumed by the system if the user specifies no operand. In the above example, INPUT is the default value.

4. The ellipsis denotes the optional occurrence of the preceding syntactical unit one or more times in succession. A syntactical unit is any combination of operand representations, commas and notational symbols enclosed in square brackets.

5. Uppercase (capital) letters must be entered by the user exactly as shown in the macro description. For example, the operation and the coded values in the operand field must always be specified in uppercase letters.

6. Commas and parentheses must be written as shown in an operand field. They are delimiters, not notational symbols.

A

Conventions

Macro description value mnemonics

Value mnemonics give the user a brief description of what the form of a particular operand should look like. The following 11 value mnemonics are used in this manual:

- relexp
- addr
- exit-addr
- addrexp
- integer
- absexp
- value
- text
- code
- symbol
- chars

In the format, only the positional operand is supplied. The particular form of the operand (value mnemonic) can be obtained from the description.

Example:

Name	Operation	Operands
[symbol]	EXAMP	name

Each keyword operand is defined by the keyword, an equals sign and a value. The particular form of the value is contained in the description of the operand, e.g. "(value)".

Example:

Name	Operation	Operands
[symbol]	EXAMPT	KEYWI=value

For each value mnemonic, one or more operand forms is permitted. For example, the value mnemonic "relexp" denotes that a relocatable expression can be written as the operand form; whereas the value mnemonic "addrexp" specifies that an explicit or implicit address must be written.

The following 11 operand forms are used:

- relocatable expression (relative expression)
- register notation
- explicit address
- implicit address
- symbol
- decimal integer
- absolute expression
- code
- text
- characters
- oplist

Fig. A.1-1 illustrates the value mnemonics and their permissible operand forms.
The operand forms are described on the pages following this table.

Value mnemonic	Relocatable expression	Register notation	Explicit address	Implicit address	Symbol	Decimal integer	Absolute expression	Code	Text	Characters
relexp	x									
absexp							x			
addr	x	x								
exit-addr		x	x	x						
addrexp			x	x						
integer						x				
value		x								
text									x	
code								x		
symbol					x					
chars										x

Fig. A.1-1: Macro convention operand forms

Note:

An X indicates that the operand form may be written.

Conventions

Relocatable expression

A relocatable expression is one whose value would change by n if the program in which it appears were relocated n bytes away from its originally assigned area of storage. All relocatable expressions must have a positive value. A relocatable expression may contain relocatable terms -- alone or in conjunction with absolute terms -- under the following conditions:

1. There must be an odd number of relocatable terms.

2. All relocatable terms but one must be paired. Pairing is described under "Absolute expression" below.

3. Any unpaired term must be directly preceded by a minus sign.

4. A relocatable term must not occur in a multiply or divide operation.

Example:

In the following examples of relocatable expressions, SAM, JOE and FRANK are in the same control section (CSECT) and are relocatable; PT is absolute.

```
SAM
SAM-JOE+FRANK
JOE-PT+5
SAM+3
```

SAM-JOE is not relocatable, because the difference between two relocatable addresses is constant.

Register notation

Register notation is written as an absolute expression enclosed in parentheses. When computed, the absolute expression must lie between 2 and 12 so that the corresponding general register can be referenced.

In the following examples of register notation, SAM and JOE are relocatable and PAL is absolute.

```
(5)        indicates general register 5
(SAM-JOE)
(PAL)
(PAL+3)
```

Explicit address

The explicit address is written in the same form as an Assembler language operand, i.e.:

```
a(b,c)
|  |  |
|  |  └──────> Base register
|  └─────────> Index register
└────────────> Displacement
```

Examples of explicit addresses:

2(0,5)
0(2,4)

Implicit address (indexed)

An implicit address is written as a symbol, optionally indexed by a specified index register.

Examples of implicit addresses:

GUPOFF
ALPMAY(4)

Symbol

The operand is written as a string of up to eight alphanumeric characters, the first of which is alphabetic. Embedded commas and blanks (spaces) in the string are not permitted. Symbols beginning with a dollar sign ("$") must not be used. The symbols beginning with this character are reserved for system use.

Examples of symbols:

LEE
BILL8SAM
DESDEB

Decimal integer

The operands may be written as a whole decimal number of up to 8 digits, e.g. 5, 31, 127 etc.

Absolute expression

An absolute expression may be an absolute term or any arithmetic combination of absolute terms. An absolute term may be an absolute symbol or any of the self-defining characters. All arithmetic operands are permitted between absolute terms.

An absolute expression may contain relocatable terms - alone or in combination with absolute terms - under the following conditions:

1. There must be an even number of relocatable terms in the expression.

2. The relocatable terms must be paired. Each pair of terms must have the same relocatability attributes, i.e. appear in the same control section of an assembly. Each pair must consist of terms with opposite signs. The paired terms need not be contiguous, e.g. RT+AT-RT, where RT is relocatable and AT is absolute.

3. A relocatable term must not occur in a multiply or divide operation.

   The pairing of relocatable terms (with opposite signs and the same relocatability attributes) cancels the effect of relocation. The values represented by the paired terms remain constant, regardless of program relocation. For example, in the absolute expression

   A-Y+X

   A is an absolute term, and X and Y are relocatable terms with the same relocatability attribute. If A equals 50, Y equals 25 and X equals 10, the value of the expression is 35. If X and Y, however, were to be relocatable by a factor of 100, their values would then be 125 and 110. However, the value of the expression would still be 35, since 50-125 +110 = 35.

An absolute expression reduces to a single absolute value.

In the following examples of absolute expressions, JOE and SAM are relocatable and defined in the same control section; BERNY and DAVE are absolute:

```
331
DAVE
BERNY+DAVE-83
JOE-SAM
DAVE+4+BERNY
```

Code

Example:

Name	Operation	Operands
[symbol]	FTBAL	scores

Code is written exactly as defined in the description of the macro instruction:

HT Half time
FT Full time

The macro instruction should be written in a program as follows:

```
SAM  FTBAL    HT
     FTBAL    FT
```

Text

A text operand is written as a string of alphanumeric characters enclosed in single quotes(''). Embedded blanks (X'40') and special characters are permitted. If a single quote or a plus sign is to be used in the string, then 2 of these characters must be supplied in each case. The text operand, including the enclosing quotes, must not exceed 255 characters, e.g.:
'AREA,PCB,132','1256'

Characters

Character operands are written as a character string. Embedded commas or blanks (X'40') are not permitted. If a single quote or plus sign is to be used in the string, 2 of these characters must be supplied in each case. The character string need not be enclosed in quotes, e.g.:

CUBTDAVE+HEINZ+JOHN+83OPMOT

Oplist operands

In several macro descriptions the operand field is specified as:

$$\text{oplist=} \begin{Bmatrix} \text{text} \\ \\ \text{addr} \end{Bmatrix}$$

This implies that a list of keyword and/or positional operands can be written as fields of a character string and that the character string itself (enclosed in single quotes) or the address of the string can be written as the oplist operand, depending on whether the "text" or "addr" form of the operand is chosen.

If oplist is presented as a character string, i.e. the "text" form was specified, the macro expansion places it in the assembled program followed by an end-of-message code, and loads a pointer to the string in register 1.

If oplist is given as an address. i.e. the "add" form was specified, the expansion places this address in register 1. In this case, the programmer must define the operands within his program and provide an end-of-message code.

To reference oplist macro operands in order to change the code, the address option of the macro is used; it is permissible to define the operand string as a series of adjacent fields, each with its own label.

The string must end with a hexadecimal 27, which serves as an end-of-message code. Any unused space within the fields of the string must be filled with blanks (X'40') to the maximum size of that field. Unlike other operand forms, all commas in an oplist operand must be written, even if no values are specified.

A typical operand string might be coded as follows:

```
OPLIST    DC C'first operand'
OPLIST1   DC C',second operand'
OPLIST2   DC C',nth operand'
          DC X'27'
```

Types of macro instruction

Most system macro instructions are of two basic types:

R-type (register) or
S-type (storage)

The name of each macro description in this publication is followed by a type indication (R, S or O). Macro instructions which are neither R-type nor S-type, referred to as other macro instructions, are denoted by (O) in their descriptions.

A

Conventions

R-type macro instructions

Address operands in R-type macro instructions are always classified as "exit-addr" or "addrexp". This arrangement allows the user to employ indexing although the addresses transferred in R-type macro instructions are properly created, i.e. the base register used for the addresses passed must contain the proper value, in order to ensure that the address refers to the desired location in virtual storage.

Example:

Assume an R-type macro instruction RTYPE, which expects an address "area" in register 1 and the "length" of that area in register 0. Its macro description would be as follows:

Name	Operation	Operands
[symbol]	RTYPE	$\left\{\begin{matrix}\text{area}\\(1)\end{matrix}\right\}, \left\{\begin{matrix}\text{length}\\(0)\end{matrix}\right\}$

Special register notation

The user's program can be written so that one or both of the operands already exist in the correct operand register when the macro instruction is executed. In this case, (1) or (0) is written as the operand. The notation (1) or (0) is referred to as special register notation. Registers (1) and (0) cannot be used in a macro instruction unless special register notation is shown in the macro instruction description.

S-type macro instructions

An S-type macro instruction is used when the number of operands to be transferred to the calling routine cannot be contained in the two operand registers. The operands are placed in an operand list whose address is transferred to the calling routine in register 1.

There are three forms of the S-type macro instruction:

1. the standard form
2. the L-form (operand list only)
3. the E-form (executable code only)

All S-type macro instructions are of the L-form and E-form unless otherwise stated under the individual descriptions.

S-type - standard form

The standard form of the S-type macro instruction generates both the operand list required by the calling routine and the linkage to that routine.

Address operands in the standard form of the S-type macro instructions are always classified as "addr". Consequently, they must not be indexed, and the user program is not responsible for maintaining the transfer registers.

Example:

Assume an S-type macro instruction called STYPE, which expects the addresses of two areas, "input" (exit-addr) and "output" (exit-addr), as well as the "length" (value) of these areas. The macro description would then be as follows:

Name	Operation	Operands
[symbol]	STYPE	input,output,length

A

S-type - L-form

The L-form macro instruction is used to create an operand list. E-form macro instructions are used to point to the operand list that is generated by the L-form macro instruction.

The Assembler recognizes an L-form macro instruction by the presence of the keyword operand MF=L in its operand field.

Because the L-form macro instruction generates only an operand list, the use of operand types which require executable codes, such as register notation, is prohibited.

There is an internal difference in the kinds of operands required in the macro description, when using the various forms of the S-type macro instruction.

If the standard form indicates operands like "addr" or "value" (i.e. register notation is allowed), it is implicitly understood that L-form macro instructions allow only operands like "relexp" and "absexp", i.e. register notation is not allowed.

Therefore, the L-form macro STYPE described above would become:

Name	Operation	Operands
[symbol]	STYPE	input,output,value,MF=L

The name field is required in the L-form because it usually becomes the label of the generated operand list and is referred to by the E-form macro instruction.

The L-form macro instruction generates the operand list at the place the macro instruction is encountered. As the L-form expansions contain no executable code, the list should be placed in the program area which contains DS (Define Storage) and DC (Define Constant).

Conventions

S-type - E-form

A generated operand list may be referred to by an E-form macro instruction. The Assembler recognizes an E-form macro instruction by the presence of the keyword operand:

$$MF=(E,\begin{cases}list\\(1)\end{cases})$$

in its operand field.

"list" specifies the location of the operand list to be used by the E-form macro instruction. If (1) is written, register 1 must be loaded with the address of the operand list before execution of the macro instruction. The symbol in the name field of an L-form macro instruction becomes the name of the operand list.
Consequently, the above E-form macro STYPE would have this format:

Name	Operation	Operands
[symbol]	STYPE	$MF=(E,\begin{cases}list\\(1)\end{cases})$

Other macro instructions

Certain system macro instructions cannot be classified as either R-type or S-type; they are referred to simply as other macro instructions, denoted by (O) in the macro descriptions.

Error codes

Macros which return error codes in registers must place the codes in the low-order byte of register 15; the leftmost three bytes are set to zero.

Zero in the low-order byte means that no error occurred. Otherwise, the error code would be a multiple of four (to facilitate indexing).

Exception:

The data management macros (service macros and access methods) place the DMS error code in the ID1ECB field of the P1FCB.
The file management macros (CATAL, ERASE, ...) use the two lowest-order bytes in register 15.

Literals

Macros may generate literals. In macros where the user is responsible for maintaining the registers for operands, he must also ensure that the literals are generated. Conversely, where the user does not have to load the registers, the literals are generated by the macros.

A.2 DMS DSECTS (DUMMY PROGRAM SECTIONS)

DSECTs of DMS tables, operand lists, FCBs and catalog entries are available to user programs by means of macros. The names and sizes of the component fields are also defined; however, the arrangement of the sections is subject to change.

The macro name and a brief description of each DMS DSECT is given in Table A.2-1. Only those DSECTs of immediate interest to users are listed.

Any user interested in the current expansion of a specific DSECT can use the following macro.

The format of the call is:

Operation	Operands
macro name	[D][,{prefix / *}]

D — This operand specifies that the macro is to generate the DSECT.

Default value: The DSECT statement is not generated.

prefix — This operand specifies the single character prefix to be added to the beginning of all labels in the DSECT.

* — This operand specifies that no prefix is to be used.

Default value: I

A

Macro name	DSECT description
IDCAT	CATALOG (CATAL) operand list
IDCE	Catalog entry
IDCEG	Catalog entry, addition for file generation groups
IDCEX	Catalog entry extension
IDCHA	CHANGE (CHNGE) operand list
IDCOP	COPY macro operand list
IDECB	UPAM event control block
IDEE	Catalog entry, extent list
IDEMS	DMS error messages This macro generates a list of EQUs describing the error messages of the DMS modules (see Appendix A.3, "DMS Error Codes")
IDERS	ERASE macro operand list
IDFCB	FCB (P1 section) Note: this DSECT describes all FCB formats
IDFST	FSTATUS (FSTAT) macro operand list
DMAIMP	IMPORT macro operand list
IDMCB	EAM control block
IDOST	File information about opened files of the user
IDPFL	FILE macro operand list
IDPFX	FILE macro operand extension
IDPPL	UPAM operand list
IDREL	RELEASE (REL) macro operand list
IDVRF	VERIF macro operand list
IDVT	Volume label entry
DMADR	RDTFT information (when the LINK operand is specified)
DMARD	RDTFT macro operand list

Table A.2-1: DMS DSECTs of interest to the user

A.3 DMS ERROR CODES

Error codes contain all the necessary information about the errors affecting programs and tasks. From them the user can derive details of the type and origin of an error, as well as what action he should take. The error codes employed in the DMS offer the following advantages.

- The abbreviated format permits the coding of many different errors.
- The coding permits identification of the source of the error.
- In the case of recoverable errors, abnormal termination can be avoided by means of self-analysis by the program.

Types of error

- DMS errors during program execution

  Error codes occurring in connection with file management macros (CATAL, CHNGE, COPY, ERASE, FILE, FSTAT, REL, VERIF) are stored in register 15 (in the 2 lowest-order bytes).

  Error codes which occur when using data management macros (service macros and access methods) are stored in the ID1ECB field of the P1FCB (displacement X'98').

  In the case of errors occurring during program execution, the DMS analyzes the stored error code (via EXLST error exits). Either the error condition can be corrected by the program, or the program is abnormally terminated and an error message issued (SYSOUT for interactive mode, SYSLST for batch mode).

- DMS command errors

  In the event of an error occurring when using DMS commands, a message is issued via SYSOUT.

  In interactive mode, the user can enter the correct command.

  In batch mode, the program branches to the next STEP command or, if none exists, to the LOGOFF command.

  The message texts for all error messages can be retrieved from MSGFL (message file) by means of the error code.
  The message texts frequently contain blanks, these being intended for information specific to the particular error.

A

DMS error codes

Error coding scheme

DMS error codes have the following structure.

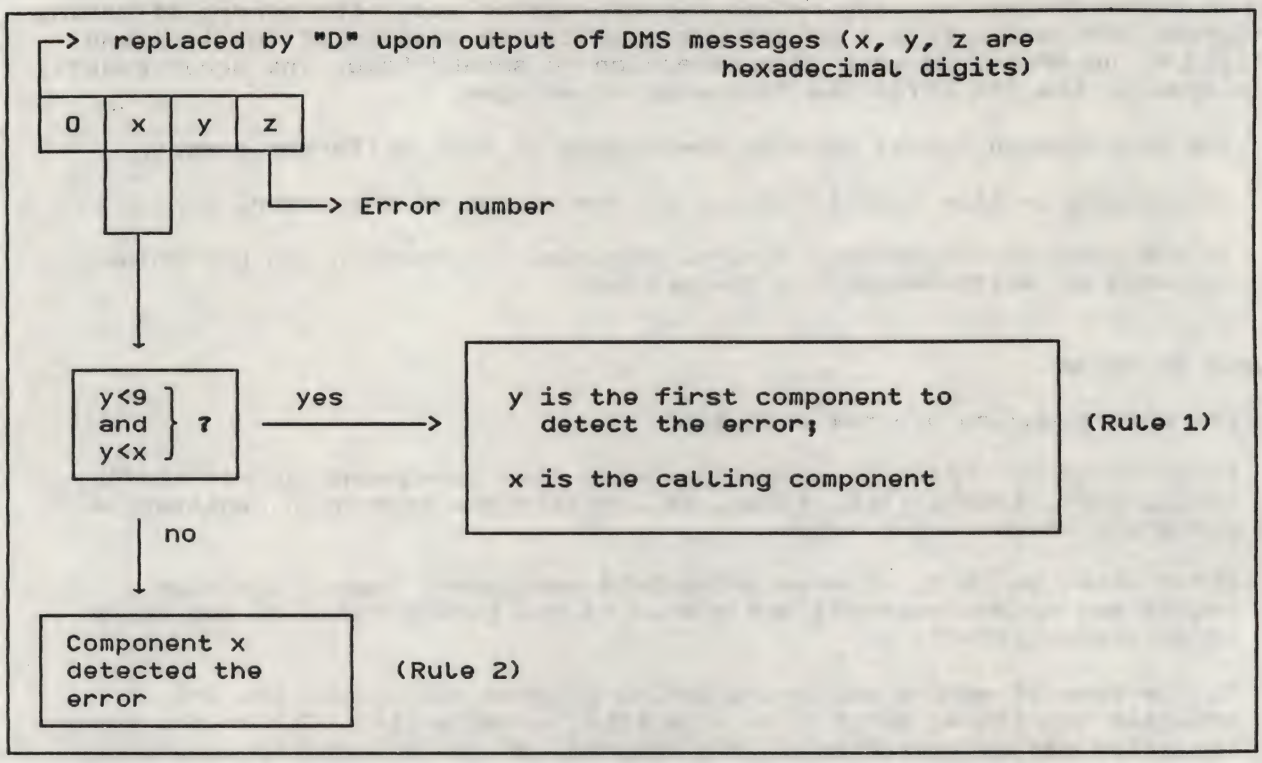


Fig. A.3-1: DMS error codes

x	y	Components
2	2	Privileged PAM (PPAM)
3	3	Catalog Manager (CMS)
4	4	Data Storage Allocator (ALLOC)
5	5	Commands with corresponding macros
6	6	Commands without corresponding macros
7		Privileged Tape Access Method (PTAM)
9		UPAM
A		ISAM
B		SAM
C		BTAM
D		OPEN
E		CLOSE

Table A.3-1: Coded values of components

DMS error codes

The order of the components identified by the hexadecimal numbers 2 (for PPAM) through E (for CLOSE) is not arbitrary. Rather it indicates that a component with a higher number can call a component with a lower number (and not vice versa).

Should an error arise in a called component, its error code is returned to the calling component and thus modified.

Examples of the generation and modification of error codes

1. One level (error in the calling component)

   Assume that a user calls UPAM and the file is not open. UPAM would place error code 0994 in the FCB and exit via $GOTO and USERERR.

2. Two levels (error in the called component)

   Assume that a user calls UPAM, which in turn calls PPAM, and PPAM detects an I/O error. In this case, PPAM returns error code 0227 to UPAM in register 15. UPAM alters this code to 0927, places it in the FCB, and exits via $GOTO to ERRADR.

List of DMS error codes

The explanations which refer to the individual error codes appear in a list.

A list of all error codes with the appropriate explanatory texts can be found at the end of this section.

This list may be printed out, either in full or in part, by means of the IDEMS macro:

Operation	Operands	Comments
IDEMS	[ALL=Y]	All error codes are generated
	[,PAM=Y]	PPAM codes only
	[,CATAL=Y]	CMS codes only
	[,ALLOC=Y]	ALLOC codes only
	[,CMDMAC=Y]	Only codes for commands with corresponding macro
	[,CMDNMAC=Y]	Only codes for commands without corresponding macro *)
	[,UPAM=Y]	UPAM codes only
	[,ISAM=Y]	ISAM codes only
	[,SAM=Y]	SAM codes only
	[,BTAM=Y]	BTAM codes only
	[,OPEN=Y]	OPEN processing codes only
	[,CLOSE=Y]	CLOSE processing codes only
	[,P=prefix]	Appears before all symbolic names in DMS messages Default value: I No letter generated for P=

*) FRS messages no longer have any significance

In the list of DMS error codes, the codes under 0x90 are missing when x > 9 (except for SAM). If such a code occurs, its constituent components must be analyzed.

A

DMS error codes

Note:

DMS error codes are also contained in the "System Messages" manual. In order to locate a DMS error code in that manual, the leading zero must be replaced by the letter D; example: 0D33 -> DD33.

Exception 1: ISAM

The ISAM access method does not conform to these rules. Instead the following always applies:

'0xyz'='0AAz', z=0...F

(z does not take all values within the range 0...F.)

Example:

If PPAM detects an error after ISAM has been called, '0AA9' is always entered in the FCB. For this error the IDEMS list will contain

SYSTEM ERROR,HARDWARE

ISAM produces this message for all 12 types of error occurring in PPAM.

Exception 2: OPEN

Under normal circumstances, the OPEN routine conforms to the formal rules regarding coding. However, when opening an ISAM file, errors may occur which cannot be explained on the basis of these rules.

Example:

If an ISAM file is opened in INPUT mode, the first PAM block is read. This block contains the highest index level.

After an I/O error, the following occurs:

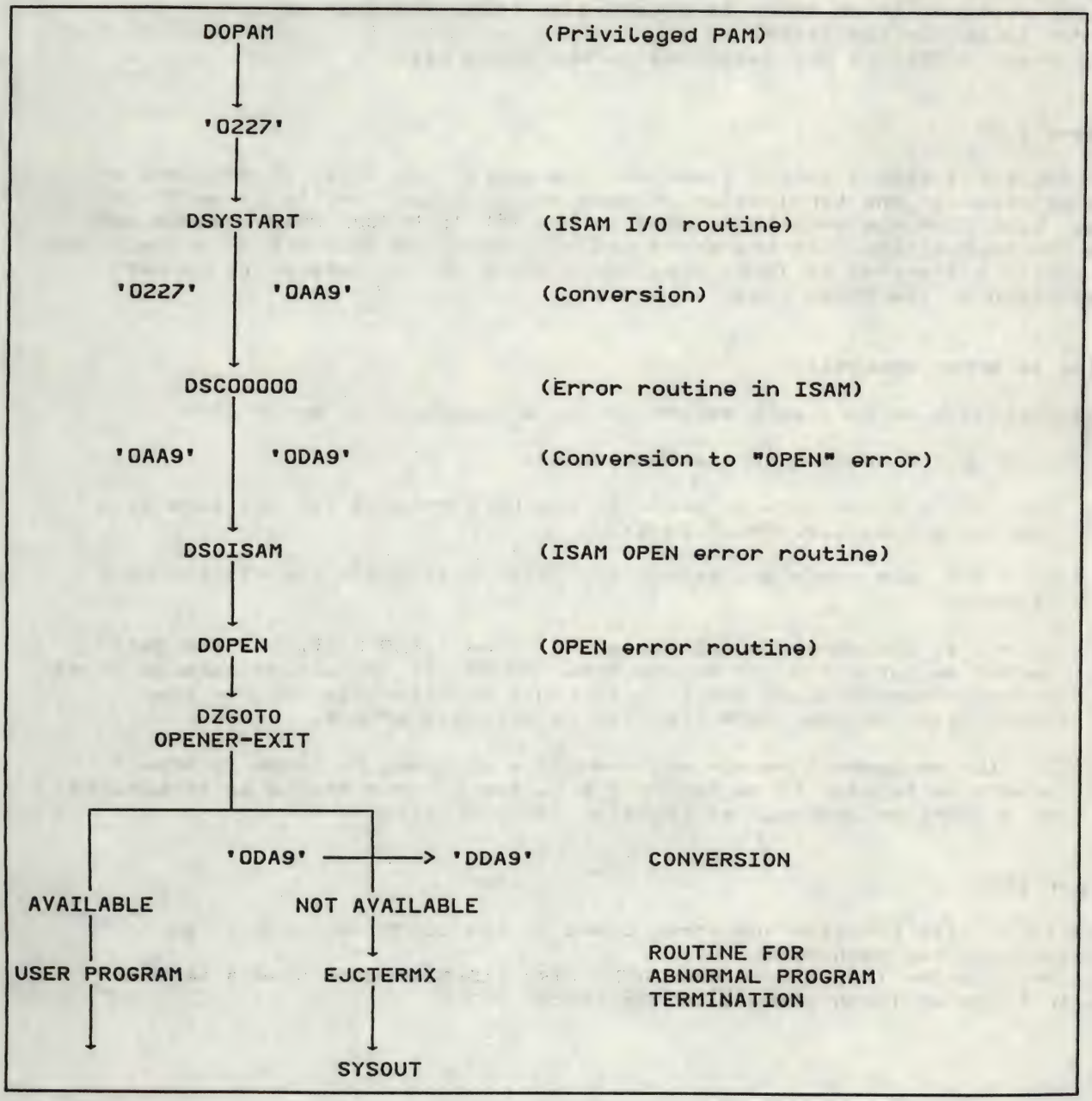


Fig. A.3-2: This error has a different meaning in the IDEMS macro; however, here it denotes an I/O error.

<u>Exception 3: CLOSE</u>

In order to terminate all outstanding I/O operations, the CLOSE routine calls the access method appropriate to the file. If an error occurs in this access method, an error code y ≥ 9 is placed in the P1FCB. In this case, rule 1 can also be used when y ≥ 9.

DMS error codes

Example 1:

When closing an ISAM file, DCLOSE (CLOSE routine) calls the ISAM-specific CLOSE routine. If an error is detected in this, the user will find the error 'OEEE' in his P1FCB.
The error 'OEEE' is **not** described in the IDEMS list.

Example 2:

DCLOSE calls ISAM a second time. Any I/O operations still in progress are terminated. If the termination of such an operation results in an error, the user will find the error code OEA9 in the file's P1FCB. Rule 2 would lead to the supposition that the error had been detected by CLOSE -- while it was actually discovered by ISAM. Like the example above, this error is **not** described in the IDEMS list.

Aids to error analysis

The following actions will assist in the analysis of an error code:

1. Look up the meaning in the IDEMS list.

2. Define the error code by means of the HELP command (at the same time inserting a D, i.e. /HELP Dxyz).

3. If y < 9, set x = y and search the IDEMS list again (or via the HELP command).

4. If y ≥ 9, the error is **not** included in the IDEMS list, try the same method as for y < 9. It may be that the result appears to make no sense, in which case analyze the program logic to determine whether the description in the IDEMS list can be accurate or not.

5. If file management macros are used in a program, register 15 should always be tested. If register 15 ≠ 0, the program should be terminated or a thorough analysis of register 15 undertaken.

Error List

The error list contains the error codes in the IDEMS macro, with an explanation for each code.
The error codes included in the IDEMS list (IDEMS macro) have a zero in their first position instead of the letter "D".

List of error codes:

```
          IDEMS ALL=Y,P=
*********      DMS ERROR MESSAGES VERSION BS2000 V6.0/000
*********
         NDQIDEMS
         IDLKG VER=801,ALIGN=C
               *,VERSION 801
***            PAM
DQPAMER  EQU   X'0200'
DQTMS    EQU   X'0221'            TOO MANY SLOTS
DQHPNIF  EQU   X'0222'            HALFPAGE NOT IN FILE
DQCSTER  EQU   X'0223'            CSTAT ERROR
DQRQMER  EQU   X'0224'            REQM ERROR
DQILLOP  EQU   X'0225'            ILLEGAL OP
DQLRCLE  EQU   X'0226'            LOGICAL ERROR RETURN CODE
DQIOERR  EQU   X'0227'            I/O ERROR
DQINVD   EQU   X'0228'            INVALID DAR INDEX
DQINVSL  EQU   X'0229'            INVALID SLOT ADDR
DQTCHIO  EQU   X'022A'            CHAINED I/O ON TAPE
DQDNSER  EQU   X'022B'            DENSITY ERROR
DQINVXL  EQU   X'022C'            INVALID EXTENTLIST
DQWTACN  EQU   X'022D'            WAIT AND CONTINGENCY REQUESTED
DQVLLST  EQU   X'022E'            VOLUME LOST
DQBRSER  EQU   X'022F'            BOURSE ERROR
         DCAIDEMS
***            CATALOG
DKEGEN   EQU   X'0300'
DKINT1   EQU   X'0301'            INTERNAL ERROR 01
DKINT2   EQU   X'0302'            INTERNAL ERROR 02
DKINT3   EQU   X'0303'            INTERNAL ERROR 03
DKERQO   EQU   X'0330'            REQM ERROR 10
DKEPARM  EQU   X'0331'            PARAMETER ERROR
DKLKERR  EQU   X'0332'            LOCK ERROR
DKENF    EQU   X'0333'            ENTRY CANNOT BE FOUND
DKERQ4   EQU   X'0334'            REQM ERROR 04 OR 0C
DKCINT   EQU   X'0334'            CMS-INTERNAL-ERROR
DKNONSR  EQU   X'0335'             NONSHARABLE FILE
DKEFR1   EQU   X'0336'            ATTEMPT TO FREE UNLOCKED BLOCK
DKEFR2   EQU   X'0337'            ILLEGAL ATTEMPT TO FREE BLOCK
DKWFNSP  EQU   X'0338'            CATALOG FILE HAS 6 OR LESS BLOCKS
*                                  REMAINING. SYSTEM ADMINISTRATOR
*                                 MUST ISSUE FILE COMMAND TO
*                                 ALLOCATE MORE SPACE FOR CATALOG
*                                 FILE=TSOSCAT. ADDITIONAL SPACE
*                                 NOT RECOGNIZED UNTIL NEXT SESSION.
DKNSPAC  EQU   X'0339'            CATALOG FILE HAS NO MORE AVAILABLE
*                                 SPACE. SYSTEM ADMINISTRATOR MUST
*                                 ISSUE FILE COMMAND TO ALLOCATE
*                                 MORE SPACE FOR CATALOG FILE=TSOSCAT.
*                                 ADDITIONAL SPACE NOT RECOGNIZED
*                                 UNTIL NEXT SESSION.
DKEF2E   EQU   X'033A'            F1 LABEL UPDATE/DELETE ERROR
DKENLK   EQU   X'033B'                 UPDATE CLEAR ERROR:
*-             LOCATE MODE=BLOCK NOT LOCKED
*-             MOVE MODE=CALLER DID NOT READ LAST UPDATE OF ENTRY
DKEBLK   EQU   X'033C'            CATALOG FILE AT MAXIMUM SIZE
DKFCD    EQU   X'033D'             TSOSCAT OR F1 LABEL BLK DESTROYED
DKFNEF1  EQU   X'033E'             NEW FILE TO BE CREATED ON PRIVATE
*                                  VOLUME ALREADY EXISTS IN F1 LABEL
*                                  AREA
DKLKSR   EQU   X'033F'            DKLOCK: FILE SECURED BY ANOTHER TASK
         DALIDEMS
```

```
***             ALLOCATOR
DRNSP    EQU    X'0441'   NO SPACE AVAILABLE - NONE WAS ALLOCATED
DRDNA    EQU    X'0442'   DEVICE NOT AVAILABLE - $AQIR ERR 04
DRPLE    EQU    X'0443'   PARAMETER LIST ERROR
DRBDC    EQU    X'0444'   CATALOG ENTRY FORMAT ERROR
DRNOP    EQU    X'0445'   PUBLIC VOLUME NOT ON LINE
DRCTF    EQU    X'0446'   CATALOG ENTRY IS FULL
DRNVOL   EQU    X'0447'   OPERATOR CANT MOUNT VOL - $AQIR ERR 14
DRNMEM   EQU    X'0448'   INSUFFICIENT MEMORY - $REQM ERR 04
DRPER    EQU    X'0449'   ALLOCATOR GAVE BAD PARAM TO REQM OR AQIR
DRINSP   EQU    X'044A'   SOME SPACE WAS ALLOCATED - EITHER NOT
*                         ENOUGH AVAILABLE OR VOLUME NOT ACCESSIBLE
DRRER    EQU    X'044B'   CAT-ID NOT FOUND OR PVS NOT AVAILABLE
DREXSPT  EQU    X'044C'   CANNOT READ/FIND USER-ID IN USER-TABLE
DRINSP2  EQU    X'044D'   NO ADDITIONAL SPACE ALLOWED FOR USER
DREXSPU  EQU    X'044E'   DEALLOCATION ERROR
DRINSP3  EQU    X'044F'         NOT USED
         DMAIDEMS
***             COMMANDS WITH MACROS
DMCMDM   EQU    X'0500'
DMCMS01  EQU    X'0501'    REQUESTED CATALOG NOT ACCESSIBLE
DMCMS02  EQU    X'0502'    REQUESTED CATALOG IN QUIET OR HOLD MODE
DMCMS03  EQU    X'0503'    MRSCAT CONTAINS INCORRECT INFORMATION
DMCMS04  EQU    X'0504'    CMS SYSTEMERROR
DMCMS05  EQU    X'0505'    MRS COMMUNICATION ERROR
*
DMALPDN  EQU    X'0511'    FILE-PRIVATE DISK POOL INCONS.-NO ALLOC.
DMCMS12  EQU    X'0512'    REQUESTED CATALOG NOT FOUND
DMCATEX  EQU    X'0513'    CATAL-CMD REJECTED IN SYSTEMEXIT
DMERAEX  EQU    X'0514'    ERASE-CMD REJECTED IN SYSTEMEXIT
DMFILEX  EQU    X'0515'    FILE- CMD REJECTED IN SYSTEMEXIT
DMALLFS  EQU    X'0516'    ERASE-ALL FILES ?
DMNOERA  EQU    X'0517'    ERASE-CMD WITHDRAWN BY CALLER
DMCOPOV  EQU    X'0518'    COPY-FILE2 EXISTS. OVERWRITE ?
DMNOCOP  EQU    X'0519'    COPY-CMD WITHDRAWN BY CALLER
DMF2EXT  EQU    X'051A'    COPY-FILE EXISTS ALREADY. CMD TERM.
DMCMUNO  EQU    X'051B'    ALL CMDS-USERID NOT IN PVS
DMCMNAC  EQU    X'051C'    ALL CMDS-USER HAS NO RIGHT ACCESS
DMCMPWD  EQU    X'051D'    ALL CMDS-LOGON PASSWORD DIFFERENCE
DMCMPDP  EQU    X'051E'    FILE-PRIVATE DISK POOL INCONSISTENT
*
DMCMNSL  EQU    X'0521'    NO SLOTS
DMCMHNT  EQU    X'0522'    HALF PAGE NOT IN FILE
DMCMCT4  EQU    X'0523'    $CSTA-ERROR 04
DMCMRQ4  EQU    X'0524'    $REQM-ERROR 04
DMCMOP   EQU    X'0525'    BAD OP-CODE
DMCMBUF  EQU    X'0526'    BUFFER PAGE BOUND
DMCMIO   EQU    X'0527'    I/O ERROR
DMCMEXT  EQU    X'0528'    NO EXTENT-LIST
DMCMBSL  EQU    X'0529'    SLOT ADDR INVALID
DMCMCSC  EQU    X'052B'    $CSTA-ERROR 0C
DMCMRQC  EQU    X'052C'    $REQM-ERROR 0C
DMCMSC1  EQU    X'052F'    $CSTA-ERROR 10
*
DMCMBFN  EQU    X'0531'    INVALID PARAM IN CMS-CALL
DMCMFIL  EQU    X'0532'    FILE LOCKED
DMCMCAT  EQU    X'0533'    FILE NOT CATALOGED
DMCMCR   EQU    X'0534'    INTERNAL CMS-ERROR
DMCMNSH  EQU    X'0535'    FILE NOT SHAREABLE
DMCMFTU  EQU    X'0536'    FRCT-CATALOG ALREADY UNLOCKED
DMCMFTI  EQU    X'0537'    $FRCT INVALID
DMCMF1L  EQU    X'053A'    F1-LAB UPD ERROR
DMCMUCE  EQU    X'053B'    UPDATE CLEAR ERROR
```

```
DMCMFMS  EQU   X'053C'    CATALOG FILE AT MAX SIZE
DMCMF1D  EQU   X'053D'    F1 LABEL DESTROYED
DMCMFEX  EQU   X'053E'    FILE EXISTS ALREADY ON F1 LABEL
DMCMFSC  EQU   X'053F'    FILE SECURED BY ANOTHER TASK
*
DMCMNSA  EQU   X'0541'    NO SPACE AVAILABLE
DMCMAAQ  EQU   X'0542'    DEVICE NOT AVAILABLE
DMCMPLE  EQU   X'0543'    PARAMETER LIST ERROR
DMCMCEE  EQU   X'0544'    CE-FORMAT ERROR
DMCMPVN  EQU   X'0545'    PUBLIC VOLUME NOT ONLINE
DMCMCEF  EQU   X'0546'    CE AT MAX. SIZE
DMCMA14  EQU   X'0547'    OPERATOR CANNOT MOUNT VOLUME
DMCMRO4  EQU   X'0548'    INSUFFICIENT MEMORY AVAILABLE
DMCMRAE  EQU   X'0549'    REQM OR AQIR ERROR
DMCMNOS  EQU   X'054A'    NOT ENOUGH SPACE,SOME WAS ALLOCATED
DMCMARQ  EQU   X'054B'    CATID NOT FOUND OR PVS NOT AVAILABLE
DMCMNUI  EQU   X'054C'    FILE-CANNOT READ/FIND USERID IN JOINFILE
DMCMSNA  EQU   X'054D'    NO ADDITIONAL SPACE ALLOWED FOR USER
DMER54F  EQU   X'054F'    CATAL-UNEXPECTED ERROR WHEN
*                                ACCESSING JOINFILE
*
DMFCTOF  EQU   X'0550'    FILE--TFT ASSOCIATED WITH AN OPEN FILE
DMFVRVC  EQU   X'0551'    FILE--VSN REPEATED FOR TAPEFILE
DMFMDNG  EQU   X'0552'    FILE--TOS MOUNT AND DEVICE NOT PRESENT
DMFAAI   EQU   X'0553'    FILE--ABS ALLOCATION ILLEGAL
DMFNFWF  EQU   X'0554'    FILE-IMPORT--NO OR INVALID FILENAME
DMFFFPC  EQU   X'0555'    FILE--FILE PREVIOUSLY CATALOGED
DMFNDFF  EQU   X'0556'    FILE--NO OR INVALID DEVICE AND
*                                   STATE=FOREIGN
DMFIMVF  EQU   X'0557'    FILE--ILLEGAL VSN
DMFFPVI  EQU   X'0558'    FILE--FOREIGN-PUBLIC VOLUME ILLEGAL
DMFIVSN  EQU   X'0559'    FILE--ILLEGAL MOUNT PARAMETER
DMFDNA   EQU   X'055A'    FILE-IMPORT--DEVICE NOT AVAILABLE
DMFVNMF  EQU   X'055B'    FILE--1ST PRIV. VOL. OF FILE NOT MOUNTED
DMFFNIF  EQU   X'055C'    FILE-CATAL--FOREIGN FILE NOT IN VTOC
DMFINAU  EQU   X'055E'    FILE--$UID NOT ALLOWED FOR UNCAT FILE
*
DMESOUT  EQU   X'0570'    ERASE--CMD NOT ALLOWED IN INTERACT. MODE
DMESDU   EQU   X'0571'    ERASE--SYSTEMFILE IS ASSIGNED TO *DUMMY
DMESNA   EQU   X'0572'    ERASE--SYSTEMFILE NOT ASS. TO A DMS FILE
DMESEM   EQU   X'0573'    ERASE--SYSTEMFILE IS EMPTY
DMESER   EQU   X'0574'    ERASE--DMS ERROR OCCURED DURING ERASE
DMESPRP  EQU   X'0575'    ERASE--DECLARATIVE PRINT/PUNCH ACTIVE
DMTEMP1  EQU   X'0579'    CATAL-FILE--INVALID PARAM FOR TEMPFILES
*
DMFVWDI  EQU   X'0590'    FILE--VOLUME WITHOUT DEVICE ILLEGAL
DMFMPV   EQU   X'0591'    FILE--MOUNT PRIVATE VOLUMES FOR FILE
DMFIV    EQU   X'0592'    FILE-IMPORT--INVALID/TOO MANY/NOT ENOUGH
*                               VSN'S
DMFAPM3  EQU   X'0593'    FILE--ABS MUST BE MULTIPLE OF 3
DMFRQO4  EQU   X'0594'    FILE-HOLD-REL--INSUFFICIENT MEMORY
DMFPNPM  EQU   X'0595'    FILE--PUB & NON-PUB MIXED
DMFDACD  EQU   X'0596'    FILE--DEVICE TYPE DISAGREES WITH C.E.
DMFAMUB  EQU   X'0597'    FILE--ABS MUST BE UNIT BOUNDARY
DMTMPSF  EQU   X'0598'    ALL CMDS--TOO MANY PARAMETERS GIVEN
DMRLNF   EQU   X'059A'    REL--LINK NOT FOUND
DMRFO    EQU   X'059B'    REL--FILE OPEN
DMFRQOC  EQU   X'059C'    FILE--INVALID REQUEST MEMORY
DMRIL    EQU   X'059D'    REL--INVALID LINK NAME
DMRISP   EQU   X'059E'    REL--INVALID PARAM/PRIVILEGED FUNCTION
DMCCPE   EQU   X'059F'    CATAL--COMMAND PARAMETER ERROR
*
DMFCPE   EQU   X'05A1'    FILE--COMMAND PARAMETER ERROR
```

A

```
DMFIDP   EQU   X'05A2'    FILE-IMPORT--INVALID DEVICE PARAMETER
DMFISP   EQU   X'05A3'    FILE--INVALID SPACE/DSPACE PARAM
DMKLERR  EQU   X'05A4'    FILE--DSPACE,DDEVICE,DVOLUME ERROR
DMCIFP   EQU   X'05A5'    CHNGE--INVALID FIRST PARAMETER
DMCISP   EQU   X'05A6'    CHNGE--INVALID SECOND PARAMETER
DMOIFP   EQU   X'05A7'    COPY--INVALID FIRST PARAMETER
DMFAQ08  EQU   X'05A8'    FILE--DEVICE TYPE NOT AVAILABLE/ILLEGAL
DMOISP   EQU   X'05A9'    COPY--INVALID SECOND PARAMETER
DMEIP    EQU   X'05AA'    ERASE--INVALID 1ST PARAMETER
DMFSNAA  EQU   X'05AB'    FSTAT-IMPORT--NO AREA ADDRESS
DMEIP2   EQU   X'05AC'    ERASE--INVALID 2ND PARAMETER
DMFSRLE  EQU   X'05AD'    FSTAT-$RLM1/$RELM ERROR
DMFSSE   EQU   X'05AE'    FSTAT-SECURITY ERROR
DMFNSA   EQU   X'05AF'    FSTAT-PASSWORD SPECIFIED,NOT SYS ADM
*
DMFAQ10  EQU   X'05B0'    FILE--DEVICE NOT AVAILABLE
DMFLK    EQU   X'05B1'    FILE--FILE CURRENTLY IN USE
DMERNFC  EQU   X'05B2'    ERASE--NO FUNCTION CODE SPECIFIED
DMEONRE  EQU   X'05B3'    ERASE--OWNER NOT REQUESTING ERASE
DMFAQ14  EQU   X'05B4'    FILE--OPERATOR CANNOT MOUNT VOLUME
DMORQ04  EQU   X'05B5'    COPY-IMPORT--$REQM ERROR 04
DMERQ04  EQU   X'05B6'    ERASE--$REQM ERROR 04
DMSI1P   EQU   X'05B7'    FSTAT--INVALID 1ST PARAMETER
DMSRQ04  EQU   X'05B8'    FSTAT-RDTFT--$REQM ERROR 04
DMSI2P   EQU   X'05B9'    FSTAT--INVALID 2ND PARAMETER
DMSI3P   EQU   X'05BA'    FSTAT--INVALID 3RD PARAMETER
DMSI4P   EQU   X'05BB'    FSTAT--INVALID 4TH PARAMETER
DMSI5P   EQU   X'05BC'    FSTAT--INVALID 5TH PARAMETER
DMORQOC  EQU   X'05BD'    COPY--$REQM ERROR OC
DMERQOC  EQU   X'05BE'    ERASE--$REQM ERROR OC
DMEPWDE  EQU   X'05BF'    ERASE--PASSWORD ERROR
*
DMSRQOC  EQU   X'05C0'    FSTAT-$REQM ERROR OC
DMORQ10  EQU   X'05C1'    COPY--$REQM ERROR 10
DMLINKO  EQU   X'05C2'    CHNGE|FILE|REL—LINK NAME IN
*                                 PARAMETER LIST=BINARY ZEROS
DMELOCK  EQU   X'05C3'    ERASE--FILE TO BE ERASED IS LOCKED
DMFMS04  EQU   X'05C4'    FILE--$MSG I/O ERROR
DMETNV   EQU   X'05C5'    ERASE--SPACE/DATA/DESTROY NOT ALLOWED
*                                FOR TAPE FILES
DMEXPDE  EQU   X'05C6'    ERASE--EXPIRATION DATE GREATER THAN
*                                CURRENT DATE
DMERLME  EQU   X'05C7'    ERASE--$RLM1/$RELM ERROR
DMFMS08  EQU   X'05C8'    FILE--$MSG ERROR
DMECAT0  EQU   X'05C9'    ERASE--CATALOG ONLY FOR PUBLIC VOLUME
DMERAQE  EQU   X'05CA'    ERASE--$AQIR ERROR
DMCFN1E  EQU   X'05CB'    CATAL--FILENAME1 ERROR
DMCFAC   EQU   X'05CC'    CATAL--STATE=NEW AND FILE CATALED/
*                                STATE=UPDATE AND FN2 CATALED
DMCFN2E  EQU   X'05CD'    CATAL--FILENAME2 ERROR
DMCFNC   EQU   X'05CE'    CATAL--STATE=UPDATE AND FILE NOT CATALED
DMCPWDE  EQU   X'05CF'    CATAL--PASSWORD ERROR
*
DMCLOCK  EQU   X'05D0'    CATAL--LOCK ERROR
DMCAAQE  EQU   X'05D1'    CATAL-VERIFY- $AQIR  ERROR
DMCCDER  EQU   X'05D2'    CATAL--RETPD SPECIFIED AND NO CR
*                                DATE IN CE
DMCRLME  EQU   X'05D3'    CATAL--$RLM1 ERROR
DMRRL04  EQU   X'05D4'    REL--$RELM ERROR
DMCHNF   EQU   X'05D5'    CHNGE--LINK NOT FOUND
DMCHRFO  EQU   X'05D6'    CHNGE--FILE IS OPEN
DMCHNVT  EQU   X'05D7'    CHNGE--TOS VOLUME TFT DOES NOT EXIST
DMFPWDE  EQU   X'05D8'    FILE--PASSWORD ERROR
```

```
DMEEAME  EQU   X'05D9'    ERASE--SYSFILE IS NOT PRIMARY
DMFNORR  EQU   X'05DA'    FILE--1ST SPACE PARAMETER MINUS AND
*                               FILENAME STARTS WITH $
DMOITP   EQU   X'05DB'    COPY--INVALID THIRD PARAMETER
DMRRLOC  EQU   X'05DC'    REL--$REQM ERROR
DMCHNLE  EQU   X'05DD'    CHNGE--NEW LINK ALREADY EXISTS
DMINVFN  EQU   X'05DE'    ERASE--INVALID FILENAME
DMFUBLF  EQU   X'05DF'    FILE--ILLEGAL USE OF BLIM/CHKPT PARAMS
*
DMALREP  EQU   X'05E0'    ERASE/FILE--FILE LOCKED DUE TO SPACE
*                                          PROBLEM
DMNOTFT  EQU   X'05E1'    RDTFT--NO TFT WITH LNK/FN
DMFNINV  EQU   X'05E2'    RDTFT--INVALID FILENAME
DMCRQ04  EQU   X'05E4'    CATAL--$REQM ERROR 04
DMEOUT4  EQU   X'05E5'    FSTAT-$EOUT ERROR 04
DMERSVL  EQU   X'05E6'    ERASE--ERROR DURING SEVER
DMERNUS  EQU   X'05E7'    IMPORT-"NUSERID" RESERVED TO TSOS
DMINVGR  EQU   X'05E8'    INVALID NAME FOR A DISCFILE
DMEOUT8  EQU   X'05E9'    FSTAT-$EOUT ERROR 08
DMINVTC  EQU   X'05EA'    FSTAT-INVALID VTOC PARAMETER
DMMRSCE  EQU   X'05EB'    MRS-COMMUNICATION ERROR
DMCRQOC  EQU   X'05EC'    CATAL--$REQM ERROR OC
DMEOUTC  EQU   X'05ED'    FSTAT-$EOUT ERROR OC
*
DMCOPF0  EQU   X'05F0'    COPY--UNCAT FILE2 BEGINS WITH USERID
DMCOPF1  EQU   X'05F1'    COPY--FILE2 CANNOT BE USED AS OUTPUT
DMCOPEA  EQU   X'05F2'    COPY--*DUMMY FILE OPENED INPUT
DMCOPEB  EQU   X'05F3'    COPY--PASSWORD REQUIRED ISSUE
*                               PASSWORD COMMAND
DMCOPEC  EQU   X'05F4'    COPY--FILENAME1=FILENAME2
DMCOPED  EQU   X'05F5'    COPY--$RELM ERROR
DMCOPEE  EQU   X'05F6'    COPY--NON STANDARD BLOCKS INPUT FILE
DMCOPEF  EQU   X'05F7'    ERASE-GEN DOES NOT EXIST,BUT INDEX
*                               IS UPDATED
DMCOPEG  EQU   X'05F8'    COPY--MSG + DMS RC FROM DMCOPY
DMER5FA  EQU   X'05FA'    PUBLIC FILE IS NOT IN HOME CATALOG
DMER5FB  EQU   X'05FB'    IMPORT: USER ID RESERVED TO TSOS
DMER5FC  EQU   X'05FC'    IMPORT: INVALID USER-ID.
DMER5FD  EQU   X'05FD'    FILE--SPACE/DSPACE PARAM ABOUT A READ ONLY
*                               FILE OR BEFORE ITS EXPIRATION DATE
DMER5FE  EQU   X'05FE'    IMPORT--NO FILE TO IMPORT
DMER5FF  EQU   X'05FF'    COPY-FILE1 IS A REMOTE FGG
         DCOIDEMS ,Y,,
***            COMMANDS WITH NO MACROS AND VERSION 7 COMMANDS
***            WITH MACROS
DMCMDNM  EQU   X'0600'
DMPIO    EQU   X'0660'    PASSWORD--INVALID OPERAND
DMDIP    EQU   X'0661'    DROP--INVALID LINKNAME
DMDLNF   EQU   X'0662'    DROP--LINK NOT FOUND
DMHIFP   EQU   X'0663'    HOLD--INVALID LINKNAME
DMHISP   EQU   X'0664'    HOLD--INVALID SECOND PARAMETER
DMSNSA   EQU   X'0668'    SPMGT--SYS ADM NOT REQUESTING FUNCTION
*
DMER670  EQU   X'0670'    SPMGT--SYNTAX ERROR IN COMMAND
DMER671  EQU   X'0671'    SPMGT--INVALID PARAM VALUE IN COMMAND
DMER672  EQU   X'0672'    SPMGT--PVS NOT AVAILABLE
DMER673  EQU   X'0673'    SPMGT--$REQM ERROR
DMER674  EQU   X'0674'    SPMGT--ALLOC REQUEST COUNTER DAMAGE
*
DMPWR04  EQU   X'0694'    PASSWORD--NO MEMORY AVAILABLE
DMPWROC  EQU   X'069C'    PASSWORD-- REQM OC
DMINVCE  EQU   X'069D'    INVALID CATALOG ENTRY
*
```

A

```
ERRD6A0  EQU   X'06A0'    PASS: RFA PROCESSING ERROR
ERRD6A2  EQU   X'06A2'    SECUR: LOCAL/REMOTE RES. MIXED
ERRD6A6  EQU   X'06A6'    CATAL--AUDIT NOT ALLOWED FOR THIS USERID
ERRD6A9  EQU   X'06A9'    CAT/COPY-GEN'S OF FGG ARE MISSING
*
DMER6B4  EQU   X'06B4'    ERASE-NO GEN TO BE ERASED
DMER6B5  EQU   X'06B5'    FILE--CE NOT UPDATED AFTER SYSTEM ERROR
DMER6B8  EQU   X'06B8'    FSTAT--INVALID OPERAND
DMER6B9  EQU   X'06B9'    ERASE-FILE-CATAL--ERROR IN SYSTEM EXIT
*                                            BASE MECHANISM
*
ERRD6C0  EQU   X'06C0'    CATAL- STATE=N & ACCESS=READ
ERRD6C1  EQU   X'06C1'    GEN PARAM. INVALID
ERRD6C2  EQU   X'06C2'    FILE LOCKED BY ANOTHER TASK
ERRD6C3  EQU   X'06C3'    INVALID FGG NAME
ERRD6C4  EQU   X'06C4'    FGG UNDEFINED
ERRD6C5  EQU   X'06C5'    FGG NAME TOO LONG
ERRD6C6  EQU   X'06C6'    ILLEGAL STATE UPDATE
ERRD6C7  EQU   X'06C7'    INCORRECT GEN.NUM.
ERRD6C8  EQU   X'06C8'    STATE=UPDATE FOR A GEN. UNALLOWED
ERRD6C9  EQU   X'06C9'    ILLEGAL FIRST/GEN/BASE/DISP
ERRD6CA  EQU   X'06CA'    ILLEGAL BASE PARAM.
ERRD6CB  EQU   X'06CB'    AREA SIZE TOO SMALL
ERRD6CC  EQU   X'06CC'    NO FILE CORRESP. TO SUPPORT/STATE
ERRD6CD  EQU   X'06CD'    FGG IS READ ONLY
ERRD6CE  EQU   X'06CE'    INVALID RETENTION PERIOD
ERRD6CF  EQU   X'06CF'    INVALID FILENAME PARAM.
*
ERRD6D0  EQU   X'06D0'    FOREIGN PRIVATE GEN. DOES NOT EXIST
ERRD6D1  EQU   X'06D1'    FGG SECURED
ERRD6D2  EQU   X'06D2'    INVALID FILE PARAM.
ERRD6D3  EQU   X'06D3'    INCORRECT USE OF FGG
ERRD6D4  EQU   X'06D4'    GENERATION CANNOT BE ERASED
ERRD6D5  EQU   X'06D5'    GROUP/FILE IS READ ONLY
ERRD6D6  EQU   X'06D6'    ERASE OF SOME FILES ON ERROR
ERRD6D7  EQU   X'06D7'    COPY--GROUP CANNOT BE COPIED INTO MEMBER
ERRD6D8  EQU   X'06D8'    GEN. HAVE NOT SAME ATTRIBUTES
ERRD6D9  EQU   X'06D9'    FSTAT-ERROR IN GEN/TYPE/SUPPORT/
*                         STATE PARAMETER
ERRD6DA  EQU   X'06DA'    CATAL-FILE--PROHIBITED MIXTURE OF
*                         GENERATIONS ON PR.DISK AND TP/PUB
ERRD6DB  EQU   X'06DB'    CATAL- STATE=F/VOLUME/DEVICE INVALID
ERRD6DC  EQU   X'06DC'    PASSWORD-PW NOT IN PW-TABLE
ERRD6DD  EQU   X'06DD'    DUP VSN
ERRD6DF  EQU   X'06DF'    ILLEGAL FSEQ/VSEQ/TSET
*
DMVIFN   EQU   X'06E0'    VERIFY-INVALID FILENAME(S)
*                                 OR SUPPORT/REPAIR PARAMETER
DMVNQFN  EQU   X'06E1'    VERIFY-FILENAME(S) NOT FULL QUALIFIED
DMVLOCK  EQU   X'06E2'    VERIFY-FILE TO REPAIR MAY NOT BE UNLOCKED
DMVNULL  EQU   X'06E3'    VERIFY-NO FILE TO REPAIR
DMVRQME  EQU   X'06E4'    VERIFY-REQM ERROR
DMVRLME  EQU   X'06E5'    VERIFY-RELM ERROR
DMVMT    EQU   X'06E6'    VERIFY-FILE TO REPAIR IS EMPTY
DMVNRP   EQU   X'06E7'    VERIFY-FILE NOT REPAIRABLE
*                               (SEE / VERIFY RESTRICTIONS)
DMVISR1  EQU   X'06E8'    VERIFY-FILENAME2 NOT CATALOGED AND
*                               FILE1 IS PRIVATE
DMVISMT  EQU   X'06E9'    VERIFY-ISAM FILE2 IS EMPTY
DMVISP1  EQU   X'06EA'    VERIFY-CANNOT PROCESS ISAM FILE1
*                               REPAIRING TERMINATED
DMVISGF  EQU   X'06EB'    VERIFY-A GARBAGE FILE WAS CREATED
*                            WITH THE UNRECOVERABLE DATA BLOCKS
```

```
DMVISP2  EQU   X'06EC'    VERIFY-ERROR ON FILE2, ISAM
*                                   REPAIRING TERMINATED
DMVISG1  EQU   X'06ED'    VERIFY-UNABLE TO PROCESS THE "GARBAGE FILE"
DMVISRP  EQU   X'06EE'    VERIFY-FILE REPAIRED ON FILE :
*                                  "REPAIR.<TSN>.<HHMMSS>"
DMVISA2  EQU   X'06EF'    VERIFY-UNABLE TO ALLOCATE ISAM FILE2
*                                 REPAIRING TERMINATED
DMVFUL   EQU   X'06F0'    VERIFY-DO YOU REALLY WANT TO FORCE UNLOCK?
DMNOUNL  EQU   X'06F1'    VERIFY-ONLY HOMELOCKS ARE RESET
DMVFULR  EQU   X'06F2'    VERIFY-FORCED UNLOCK REQUESTED BY TSOS
DMVIUNL  EQU   X'06F3'    VERIFY-FILE UNLOCKED
DMVIREP  EQU   X'06F4'    VERIFY-FILE REPAIRED
DMINVF2  EQU   X'06F5'    VERIFY-PARAMETERS RESTRICTED TO SYSADM
*
DMER6F6  EQU   X'06F6'    IMPORT/COPY: INVALID KEYWORD
DMER6F7  EQU   X'06F7'    IMPORT/COPY: INVALID PARAM
DMER6F8  EQU   X'06F8'    DMERASE CATALOG PARAM MISSING
DMER6F9  EQU   X'06F9'    DMERASE FN AND VOLUME MISSING
DMER6FA  EQU   X'06FA'    CATAL-STATE=NEW OR FOREIGN AND
*                               FILENAME2 PRESENT
DMER6FB  EQU   X'06FB'    CATAL-EXEC-PW NOT ALLOWED FOR FGG
DMER6FC  EQU   X'06FC'    IMPORT-UNEXPECTED I/O ERROR
DMER6FD  EQU   X'06FD'    FILE-INPUT PARAM LIST INVALID
DMC00FE  EQU   X'06FE'    RFA BCAM CONNECTION ERR
DMC00FF  EQU   X'06FF'    RFA BCAM CONNECTION INOP
*
DMEROK   EQU   X'0800'    ERASE-FILE ERASED
DMERERR  EQU   X'0801'    ERASE-ERASE ERROR
DMVFREP  EQU   X'0802'    VERIFY-FILE REPAIRED ON FILE2
DMVFERR  EQU   X'0803'    VERIFY-ERROR OCCURED WHILE
*                                REPAIRING THE FILE
DMERRES  EQU   X'0806'    VERIFY-ERROR ON RESERVE
         NDWIDEMS
         IDLKG VER=831,ALIGN=C
               *,VERSION 831
***            PTAM
DWPTAME  EQU   X'0700'
DWRQMER  EQU   X'0771'            REQUEST MEMORY ERROR
DWINVD   EQU   X'0772'            INVALID DAR IDENTIFIER
DWCSTER  EQU   X'0773'            CSTAT ERROR
DWNEXST  EQU   X'0774'            WANTED TAPE POSITION DOESN'T EXIST
DWILLOP  EQU   X'0775'            ILLEGAL OP-CODE
DWLRCLE  EQU   X'0776'            LOGICAL ERROR RETURN CODE
DWIOERR  EQU   X'0777'            I/O ERROR
DWNAC    EQU   X'0778'            WAIT/CHECK NOT ALLOWED
DWVLLST  EQU   X'0779'            VOLUME LOST
DWBRSER  EQU   X'077A'            BOURSE ERROR
DWWRE    EQU   X'077B'            WRITE ERROR
         DUPIDEMS
***            NONPRIVILEGED PAM
DQNPAME  EQU   X'0900'
DQHFPG   EQU   X'0922'            HALF PAGE NOT IN FILE
DQNFCBG  EQU   X'0990'  NO FCB ADDRESS GIVEN
*                       (OR HIGH ORDER BYTE OF FCB ADDRESS NOT 0,
*                       OR EXECUTE FORM OF A PAM MACRO OCCURS
*                       WITHIN A PARAMETER LIST CHAIN
DQWTIF   EQU   X'0991'            WRITE TO INPUT FILE
DQNBAG   EQU   X'0992'            NO BUFFER ADDRESS GIVEN
*                       (OR HIGH ORDER BYTE OF BUFFER ADDR ILLEGAL)
DQBNIAS  EQU   X'0993'            BUFFER NOT IN VIRTUAL MEMORY
DQFNOAP  EQU   X'0994'            FILE NOT OPEN
DQNILOP  EQU   X'0995'            ILLEGAL OP
DQNCADG  EQU   X'0996'            NO CHECK ADDR GIVEN
```

A

```
DQIOEPO  EQU   X'0997'            I/O ERROR ON PREVIOUS OP
DQILLEN  EQU   X'0998'            ILLEGAL LENGTH
DQIPLTY  EQU   X'0999' INVALID PAM PARAMETER LIST TYPE
DQIHPN   EQU   X'099A' ILLEGAL HALFPAGE NUMBER
*                      (HIGH ORDER BYTE OF HP FIELD ILLEGAL OR
*                      HALFPAGE NUMBER NOT GREATER THAN 0)
DQRTDF   EQU   X'099B'            READ TO DUMMY FILE
DQIREQN  EQU   X'099C' ILLEGAL REQNO PARAMETER
*                      (NEGATIVE, 0, OR TOO LARGE)
DQIKFLD  EQU   X'099D' ILLEGAL HIGH ORDER BYTE IN KEYFLD PARAM
DQKFNVM  EQU   X'099E' KEYFLD NOT IN VIRTUAL MEMORY
DQIC5A   EQU   X'099F' INSUFFICIENT CLASS V WORKSPACE AVAILABLE
DQPLNWA  EQU   X'09A0' PARAMETER LIST ADDRESS NOT WORD ALIGNED
DQPLNVM  EQU   X'09A1' PARAMETER LIST NOT IN VIRTUAL MEMORY
DQTME    EQU   X'09A2' TOO MANY ELEMENTS IN PARAMETER LIST CHAIN
*                      (OR PARAMETER LIST CHAIN LINKED IN A CIRCLE)
DQF1NWA  EQU   X'09A3' FCB ADDRESS NOT WORD ALIGNED
DQF1NVM  EQU   X'09A4' FCB NOT IN VIRTUAL MEMORY
DQNVFCB  EQU   X'09A5' FCB ADDRESS IN PAM PARAMETER LIST NOT VALID
*                      (FILE NOT OPENED, FCB HAS BEEN OVER-
*                      WRITTEN, GIVEN ADDRESS IS NOT THE ADDRESS
*                      OF AN FCB, ETC.)
DQNKFMK  EQU   X'09A6' KEY FIELD NOT SPECIFIED & MULTIPLE KEYS WISHED
DQEVDUP  EQU   X'09A7' SAME EVENT IDENTIFIER FOR TWO OUTSTANDING I/O'S
DQSYASC  EQU   X'09A8' IMPOSSIBLE COMBINATION OF SYNCHRONOUS
*                      AND ASYNCHRONOUS PARAMETERS
DQRQBNF  EQU   X'09A9' REQUEST BLOCK NOT FOUND
DQEVINE  EQU   X'09AA' EVENT IDENTIFIER NOT ENABLED
DQSETMK  EQU   X'09AB' MULTIPLE PAGES ON A SETL OPERATION
DQNFRQB  EQU   X'09AC' NO MORE FREE REQBK AVAILABLE
DQNPDLS  EQU   X'09B0' DEADLOCK
DQNALA   EQU   X'09B1' NOT ALL REQUESTED LOCKS AVAILABLE
DQUNNL   EQU   X'09B2' ATTEMPT TO UNLOCK A PAGE  NOT LOCKED
DQLNAL   EQU   X'09B3' ATTEMPT TO LOCK A PAGE ALREADY LOCKED
DQTML    EQU   X'09B4' ATTEMPT TO ACQUIRE TOO MANY LOCKS
DQIC3A   EQU   X'09B5' INSUFFICIENT CLASS III WORKSPACE AVAILABLE
DQIC4A   EQU   X'09B6' INSUFFICIENT CLASS IV WORKSPACE AVAILABLE
DQUP9F0  EQU   X'09F0'            ILLEGAL PPL CHAIN
DQUPOFE  EQU   X'09FE'            RFA BCAM CONNECTION ER
DQUPOFF  EQU   X'09FF'            RFA BCAM CONNECTION INOP
         DISIDEMS
***            ISAM
DSISAME  EQU   X'0A00'
DSISRQ1  EQU   X'0A01'            ISREQ:NO ISAM PAGE LOCK
DSISRQ2  EQU   X'0A02'            ISREQ:ISAM LOCK ON ANOTHER FILE
DSNULL   EQU   X'0AA0'            NULL ISAM FILE BEING OPENED
*                                 INOUT/EXTEND
DSSOFT   EQU   X'0AA1'            SYSTEM ERROR, SOFTWARE
DSIO     EQU   X'0AA2'            SYSTEM ERROR, INPUT/OUTPUT
DSUSER   EQU   X'0AA3'            USER ERROR, ILLOGICAL USE OF AN
*                                 ACTION MACRO OR ILLEGAL ACTION
DSSPACE  EQU   X'0AA4'            INSUFFICIENT SPACE FOR DATA
DSISPER  EQU   X'0AA5'            INSUFFICIENT SPACE FOR INDEX
DSBADF1  EQU   X'0AA6'             INVALID P1 FCB ADDR GIVEN TO ISAM
*                                  BY USER ON SVC CALL
DSVALER  EQU   X'0AA7'            BUFFER POINTER VALIDATION ERROR
DSNOFND  EQU   X'0AA8'            SPECIFIED RECORD COULD NOT BE FOUND
DSHARD   EQU   X'0AA9'            SYSTEM ERROR, HARDWARE
DSPGLOK  EQU   X'0AAA'            PAGE IS LOCKED
DSSEQ    EQU   X'0AAB'            RECORD OUT OF SEQUENCE
DSSPCZR  EQU   X'0AAC'            2ND ALLOCATION COUNT =0
DSEOF    EQU   X'0AAE'            END OF FILE
DSDUPKY  EQU   X'0AAF'            DUPLICATE KEY
```

```
DSISOFE  EQU   X'0AFE'          RFA BCAM CONNECTION ER
DSISOFF  EQU   X'0AFF'          RFA BCAM CONNECTION INOP
         DSAIDEMS
***            SAM
DISAMER  EQU   X'0B00'
DITMS    EQU   X'0B21'          TOO MANY SLOTS
DIHPNIF  EQU   X'0B22'          HALF PAGE NOT IN FILE
DICST04  EQU   X'0B23'          PAM CSTAT ERR 04
DIRQM04  EQU   X'0B24'          PAM REQM ERR 04
DIILLOP  EQU   X'0B25'          PAM ILLEGAL OP
DIBNOP   EQU   X'0B26'          PAM BUFF CROSSES PAGE
DIIOERR  EQU   X'0B27'          PAM IO ERR
DINOPDT  EQU   X'0B28'          PAM NO PDT ADDR
DIINVSL  EQU   X'0B29'          PAM INVALID SLOT ADDR
DICSTOC  EQU   X'0B2B'          PAM CSTAT ERR OC
DIRQMOC  EQU   X'0B2C'          PAM RQM ERR OC
DICST10  EQU   X'0B2F'          PAM CSTAT ERR 10
DIWCST4  EQU   X'0B73'          PTAM CSTAT ERR 04
DIWILOP  EQU   X'0B75'          PTAM ILLEGAL OP
DIWBNOP  EQU   X'0B76'          PTAM BUFF CROSSES PAGE
DIWIOER  EQU   X'0B77'          PTAM IO ERR
DIWCSTC  EQU   X'0B7B'          PTAM CSTAT ERR OC
DIWCSTF  EQU   X'0B7F'          PTAM CSTAT ERR 10
DIBLKCK  EQU   X'0B80'        ENSVC147- BLOCK COUNT CHECK FAILS
DIFAI    EQU   X'0BB0'          FCB ADDRESS INVALID
DINOEOF  EQU   X'0BB1'          EOF ADDR OMITTED FROM EXIT LIST
DINONSP  EQU   X'0BB2'          NOSPACE OMITTED FROM EXIT LIST
DISTL1E  EQU   X'0BB3'          SETL CANNOT BE PERFORMED
DISTL2E  EQU   X'0BB4'          SETL BUFF NR EXCEEDS FILE LIMITS
DISTL3E  EQU   X'0BB5'          SETL RCD NR NOT IN BUFFER
DILLOG1  EQU   X'0BB6'          ACTION MACRO NOT PERMITTED
DINWLER  EQU   X'0BB7'          WLERR/ERROPT OMITTED FROM EXIT LIST
DINEROP  EQU   X'0BB8'          ERRDPT OMITTED FROM EXIT LIST
DINDATA  EQU   X'0BB9'          KEY VARIES FROM FLID
DINTPAM  EQU   X'0BBA'          SAM HAS READ NON-PAM BLOCK
DIURWLR  EQU   X'0BBB'          USER HAS RECD WR LGTH ERR
DINBLIM  EQU   X'0BBC'          EOV BEFORE BLIM CONDITION OCCURED
DISAOFE  EQU   X'0BFE'          RFA BCAM CONNECTION ER
DISAOFF  EQU   X'0BFF'          RFA BCAM CONNECTION INOP
         DBTIDEMS
***            BTAM
DPBTAME  EQU   X'0C00'          EOF FOR *DUMMY FILE
DPNFCBG  EQU   X'0C90'          NO FCB
DPWTIF   EQU   X'0C91'          WRITE TO INPUT FILE
DPNBAG   EQU   X'0C92'          NO BUFFER ADDRESS GIVEN
DPBNIAS  EQU   X'0C93'          BUFFER NOT IN VIRTUAL MEMORY
DPCST04  EQU   X'0C94'          CSTAT ERROR 04
DPILLOP  EQU   X'0C95'          ILLEGAL OP
DPNCADG  EQU   X'0C96'          NO CHECK ADDR GIVEN
DPIOEPO  EQU   X'0C97'          I/O ERROR ON PREVIOUS OP
DPIOERR  EQU   X'0C98'          I/O ERROR
DPBCPB   EQU   X'0C99'          BUFFER CROSSES PAGE BOUNDARY
DPFNO    EQU   X'0C9A'          FILE NOT OPEN
DPRTDF   EQU   X'0C9B'          READ TO DUMMY FILE
DPCSTOC  EQU   X'0C9C'          CSTAT OC
DPCST10  EQU   X'0CA0'          CSTAT 10
DPRQM04  EQU   X'0CB4'          REQM 04
DPRQMOC  EQU   X'0CBC'          REQM OC
DPBTOFE  EQU   X'0CFE'          RFA BCAM CONNECTION ER
DPBTOFF  EQU   X'0CFF'          RFA BCAM CONNECTION INOP
         DOPIDEMS
******** OPEN  ******************
DOPENER  EQU   X'0D00'
```

A

```
DONOOPN  EQU   X'0D10'          OPEN RESTRICTED BY SYSTEM ADMIN.
DOCKPER  EQU   X'0D12'          ERROR DURING CHECKPOINT
DOEXERR  EQU   X'0D15'          ERROR DURING RZ-EXIT ROUTINE
*
DOILPOS  EQU   X'0D79'          ILLEGAL␣TM␣COUNTER
*
DOFCBO   EQU   X'0D90'          FCB ACTIVE/TFT ACTIVE/TFT TOS
DOPASSE  EQU   X'0D91'          NONE/INVALID PASSW.FOR PROTECT.FILE
DOEXPDE  EQU   X'0D92'          EXPIRATION DATE NOT SATISFIED
DONVSNM  EQU   X'0D93'          1ST PRIVATE VSN NOT MOUNTED
DOTSTIN  EQU   X'0D94'          INVALID USE OF TSET OR FSEQ
DOPARME  EQU   X'0D95'          IOAREA ERROR
DOBFORO  EQU   X'0D96'          OUTPUT/OUTIN SPEC. FOR FOREIGN TAPEF.
DOAQE    EQU   X'0D97'          $AQIR ERROR
DOSINOE  EQU   X'0D98'          SINOUT:BUT FILE NOT:TAPE/BTAM/FOREIGN
DOLOCKE  EQU   X'0D99'          FILE IS LOCKED
DONOTOP  EQU   X'0D9A'          OPEN INPUT, BUT FILE IS EMPTY
DOWRLA   EQU   X'0D9B'          $REQM OR $RELM ERROR
DOTVINV  EQU   X'0D9C'          TVSN OR VSEQ SPEC.WITH DISK FILE
DOWNVSN  EQU   X'0D9D'          NO VSN (OR NO EXTENTS) IN CAT ENTRY
DONDTCE  EQU   X'0D9E'          DEVICE TYPE NOT PRESENT IN CATALOG
DOWRQA   EQU   X'0D9F'          FCB ADDRESS INVALID
*
DOVSNNC  EQU   X'0DA1'          VSN DON'T MATCH
DONOVL1  EQU   X'0DA2'          VOL1 NOT FOUND
DODAFAL  EQU   X'0DA3'          WRONG OVERWRITE ACCESS
DOHDRCT  EQU   X'0DA4'          HDR1 OR HDR3 CONTENT (TO CONCOLE)
DONOHDR  EQU   X'0DA5'          DISPLAY WRONG FILE IDENT. TO CONSOLE
DOVSNOP  EQU   X'0DA6'          VSEQ AND OPEN NOT COMPATIBLE
DOYETOP  EQU   X'0DA8'          VOLUME CURRENTLY OPEN
DONODEV  EQU   X'0DA9'          NO FREE DEVICE EXISTS FOR MOUNTING
DOIOBOT  EQU   X'0DAA'          STATIC INOUT WITH BOT
DODZGOE  EQU   X'0DAB'          DZGOTO ENTRANCE INVALID
DODZREE  EQU   X'0DAC'          DZRET ENTRANCE INVALID
DORELER  EQU   X'0DAD'          REL MACRO ERROR
DOFCBTE  EQU   X'0DAE'          FCBTYPE IN COMPLETED P1 FCB
*                               INCONSISTENT WITH FCBTYPE IN CE
DOOTEQZ  EQU   X'0DAF'          OPEN TYPE IN P1 FCB EQUAL TO ZERO
*
DOOTIAM  EQU   X'0DB0'          OPENTYPE INVALID FOR ACCESS-METHOD
DIXTOE1  EQU   X'0DB1'          INVALID SPACE (<BLKSIZE) FOR SAM FILE
DIXTOE2  EQU   X'0DB2'          NO ALLOCATION POSSIBLE (SAM/ALLOC.)
DILRTNE  EQU   X'0DB3'          UNABLE TO LOCATE LOGICAL ROUTINE
DIRPE1   EQU   X'0DB4'          CANNOT PROCESS REV IPT FILE
DIILOGD  EQU   X'0DB5'          NSTD BLOCKS SPEC ON RA DEV
DOBADDV  EQU   X'0DB6'          FCBTYPE=ISAM AND FILE RESIDES ON TAPE
DOISRFE  EQU   X'0DB7'          FCBTYPE=ISAM AND RECFORM=U/RECFORM=F
*                               AND RECSIZE=0 IN FCB/RECFORM IN FCB
*                               NOT EQUAL TO RECFORM IN CATALOG ENTRY
DOISBER  EQU   X'0DB8'          BLK TYPE NOT STD OR BLKSIZE>(STD,16)
DOIORER  EQU   X'0DB9'          ISAM/SAM IOREG NOT BETWEEN 2-12 OR
*                               SAM VARBLD OR RECSIZE REG NOT
*                               BETWEEN 2-13
DOISBNE  EQU   X'0DBA'          FCBTYPE=ISAM AND BUFFER SIZE IN P1
*                               FCB NOT EQUAL TO BUFFER SIZE IN ISAM
*                               LABEL AREA
DOTVSQF  EQU   X'0DBB'          VOL SEQ # ERROR
DOISPAD  EQU   X'0DBC'          PAD VALUE INVALID FOR BLKS/RECS PARAM
DOISRRE  EQU   X'0DBD'          FCBTYPE=ISAM AND RECSIZE IN P1 FCB
*                               > RECSIZE IN ISAM LABEL
DOOPNSE  EQU   X'0DBE'          SYSTEM ERROR-CONDITION OCCURRED IN
*                               DOPEN WHICH IS POSSIBLE ONLY IF
*                               SYSTEM IS NOT FUNCTIONING PROPERLY
```

```
DOWVTFT  EQU   X'0DBF'        #VSNS IN TFT NOT EQUAL #VSNS IN CE
*
DOISKK   EQU   X'0DC0'        ISAM WITH RECFORM=F AND KEYLEN+
*                             (VALLEN+LOGLEN) + KEYPOS > RECSIZE
DONULLI  EQU   X'0DC1'        FCBTYPE=ISAM,OPEN TYPE=INPUT AND
*                             FILE TO BE OPENED IS NULL
DIOPER1  EQU   X'0DC2'        RECFORM/RECSIZE NULL OR IN ERROR
DOF2BIG  EQU   X'0DC3'        KEYLEN+VALLEN+LOGLEN>255
DIOPER2  EQU   X'0DC4'        BLKSIZE CANNOT BE CHANGED ON PAM
DOFDIFF  EQU   X'0DC5'        ISAM: OPEN=INOUT, VALLEN/LOGLEN/VAL-
*                             PROP NOT EQUAL TO CATALOG VALUE
DOFRSE   EQU   X'0DC6'        RECON/RESET/OPT BITS SET FOR OUTIN/
*                             OUTPUT
DOLGTHO  EQU   X'0DC7'        ISAM OPEN:  KEYLEN OR KEYPOS IS ZERO
DOOFRSM  EQU   X'0DC8'        PAM-MOVE MODE ERROR IN FRS
DOCFNT   EQU   X'0DC9'        CODED FILE NOT A TAPE FILE
DOCBCS   EQU   X'0DCA'        ILLEGAL BLKSZ ON CODED TP
DOIRCVS  EQU   X'0DCB'        ILLEGAL RECSZ ON CODED SAM V TP
DOIBCVS  EQU   X'0DCC'        ILLEGAL BLKSZ ON CODED SAM V TP
DOANSOC  EQU   X'0DCD'        A(TRTAB) NOT SPEC ON OWNCODE TP
DOUNLFD  EQU   X'0DCE'        FILE CODE INCOMPATIBLE WITH TAPECODE
DOTPLBR  EQU   X'0DCF'        UNDEFINED LABEL FIELD
*
DOPCRL   EQU   X'0DD0'        UNABLE TO OPEN DUE TO SPEC CAT LOCKS
DOPNCER  EQU   X'0DD1'        FILE NOT CLOSED-USE VERIFY TO REPAIR
DOILRES  EQU   X'0DD2'        ILLEGALE RESPONSE TO MSG 'DDE4'
ERRDDD3  EQU   X'0DD3'        ATTEMPT TO OPEN A GROUP INDEX (FGG)
DOBTMNA  EQU   X'0DD4'        ILLEGAL USE OF BTAM / PAM
DORVUIN  EQU   X'0DD5'        RECFORM=U OR V AND LABTYP NOT OK
DOWVMOU  EQU   X'0DD6'        WRONG FILE SET IDENTIFICATION
DOACCFA  EQU   X'0DD7'        ACCESS FAILURE
DOHDRNO  EQU   X'0DD8'        NO STD LABEL FOUND
DOUSLER  EQU   X'0DD9'        USER REPORTS ERROR FOR NSTD LABEL
DOPSWDI  EQU   X'0DDA'        PASSWORD DIFFERENT
DOFSQIN  EQU   X'0DDB'        INVALID FILE SEQUENCE NUMBER
DOFSQNF  EQU   X'0DDC'        ILLEGAL BLP PARAM
DOTMFNS  EQU   X'0DDD'        TM DETECTED BUT NOT SPECIFIED
DOWMNT   EQU   X'0DDE'        MOUNT MSG (WARNING)
DOWRFSQ  EQU   X'0DDF'        WRONG FSEQ
*
DOFUBLO  EQU   X'0DE0'        ILLEGAL USE OF BLIM PARAM
DOEOVEF  EQU   X'0DE2'        EOV ASSUMED AS EOF
DOPREMT  EQU   X'0DE5'        PRE_MOUNT
DOBKFAL  EQU   X'0DE6'        BLOCK COUNT FAILURE
DOVOLOC  EQU   X'0DE8'        VOL SWITCH OCCURS (SAM)
DOEOTMF  EQU   X'0DE9'        EOF/EOV NOT FOUND : TM FOUND
DOEORCF  EQU   X'0DEA'        EOF/EOV NOT FOUND : RECORD FOUND
DOEO2TF  EQU   X'0DEB'        EOF/EOV NOT FOUND : 2 TM FOUND
DOEOBCI  EQU   X'0DEC'        EOF/EOV : BLOCK COUNT INCORRECT
DOOFSEQ  EQU   X'0DED'        INV FSEQ# AT OPEN
*
DOQTMRQ  EQU   X'0DF0'        TOO MANY PAM REQUESTS
DOQRQER  EQU   X'0DF1'        NOT ENOUGH CLASS 4 OR 5 MEMORY AVAIL.
DOQTSER  EQU   X'0DF2'        TASK SYNCHRONISATION COULDN'T
*                             ACQUIRE THE REQUESTED LOCKS
DOQNPSU  EQU   X'0DF3'        ATTEMPT TO OPEN A NON-PAM FILE PAM SU
DOQTPSU  EQU   X'0DF4'        ATTEMPT TO OPEN A TAPE FILE PAM SU
DONOHME  EQU   X'0DF5'        FILE NOT ON HOME PUBLIC VOLUME
DOWLABI  EQU   X'0DF6'        WRONG LABEL FORMAT OPEN INPUT
DOWLABO  EQU   X'0DF7'        WRONG LABEL FORMAT OPEN OUTPUT
DIMVDEV  EQU   X'0DF8'        MOUNTED VSN NOT = EXPECTED VSN
DIODF9   EQU   X'0DF9'        INVALID BUFOFF
DINDINC  EQU   X'0DFA'        WRONG STD LABEL VERSION USED (DIN)
```

A

```
DONVOLU  EQU   X'0DFB'          'ACKNOWLEDGE VSN ...' (TO CONSOLE)
DOILVSN  EQU   X'0DFC'          ILLEGAL VSN
DOBYPAS  EQU   X'0DFD'          BYPASS FUNCTION CALLED
DOACFOR  EQU   X'0DFE'          ACCESS FORBIDDEN
DOIGNNA  EQU   X'0DFF'          IGNORE NOT ALLOWED
         DCLIDEMS
******** CLOSE ************
*
DCLOSER  EQU   X'0E00'
DCLEOVX  EQU   X'0E10'          EOV RESTRICTED BY RZEXIT-ROUTINE
DCLEOVER EQU   X'0E15'          EOV: ERROR IN RZEXIT ROUTINE
DCLIOERR EQU   X'0E77'          CLOSE I/O ERROR
DCLFCBO  EQU   X'0E90'          INACTIVE FCB
DCLCMSE  EQU   X'0E91'          CATAL ERROR
DCLVSNM  EQU   X'0E93'          INCORR VOLUME MOUNTED
DCWRL04  EQU   X'0E94'          RELM ERROR 04
DCPCIOE  EQU   X'0E95'          I/O ERROR(S) WHILE WAITING ONE
*                               OR MORE UNWAITED I/O(S)
DCLWPOS  EQU   X'0E96'          CANNOT WRITE EOF LABELS, LAST OP
*                               NOT WRITE (TAPE POSITION !)
PCLERROR EQU   X'0E98'          ERROR CODE PSEUDO CLOSE
DILEFCB  EQU   X'0E99'          INVALID FCB ADDRESS
DCLRLA   EQU   X'0E9A'          RELM ERROR
DCWRLOC  EQU   X'0E9C'          RELM ERROR 0C
DCLRQA   EQU   X'0E9F'          REQM ERROR
DCLFEOV  EQU   X'0EE0'          DOCTRLP- EOF AT FEOV HANDLING
DCVREV   EQU   X'0EE1'          DOCTRLP- OPEN-TYPE=REVERSE
DCVNOMT  EQU   X'0EE2'          DOCTRLP- CANNOT MOUNT TAPE
DCLISERR EQU   X'0EEE'          ISAM-CLOSE-ERROR
DCOCOFE  EQU   X'0EFE'          RFA BCAM CONNECTION ER
DCOCOFF  EQU   X'0EFF'          RFA BCAM CONNECTION INOP
```

A.4 SIGNIFICANCE OF THE SENSE BYTES FOR DIFFERENT DEVICES

The symbols x and O in column 1 have the following significance:

x If the condition is satisfied, bits 2^7 and 2^5 in byte 35 of the CCB are set to 1 by the Control System.

O If the condition is satisfied, bits 2^7 and 2^6 in byte 35 of the CCB are set to 1 by the Control System.

Card reader - sense byte 1

x 2^7	Read error
x 2^6	Service request not honored
O 2^5	
x 2^4	Invalid punch code
O 2^3	Hopper empty
O 2^2	
O 2^1	
x 2^0	Illegal operation

Card punch - sense byte 1

x 2^7	Punch compare error
x 2^6	Punch memory parity error
O 2^5	
x 2^4	Transmission parity error
O 2^3	
O 2^2	
O 2^1	
x 2^0	Illegal operation

Printer - sense byte 1

x 2^7	Channel 12 punched
x 2^6	Channel 9 punched
O 2^5	
x 2^4	Print line incomplete (243) or parity error (4247)
O 2^3	
O 2^2	
O 2^1	
x 2^0	Illegal operation

Magnetic tape - sense byte 1

x 2^7	Parity error
x 2^6	Service request not honored
O 2^5	Data block greater than count
x 2^4	Transmission parity error
x 2^3	Previous block less than count
O 2^2	BOT/EOT marker
O 2^1	Tape mark
x 2^0	Illegal operation, invalid control byte, or missing write-enable ring

A

Sense bytes

Magnetic disk - sense byte 1

x 2^7	Read error
x 2^6	Service request not honored
x 2^5	Seek check
x 2^4	Transmission parity error
x 2^3	Track check
x 2^2	Automatic head switching error
0 2^1	End of file
x 2^0	Illegal operation

Magnetic disk - sense byte 2

x 2^7	Count fields read error
x 2^6	Overflow incomplete
x 2^5	Missing start-of-block marker
x 2^4	File protect
x 2^3	Not found
x 2^2	Invalid command sequence
x 2^1	End of cylinder
0 2^0	End of track

A.5 SHAREABLE PRIVATE DISK

A complete description of this function can be found in the "MSCF Multiprocessor System" manual.

The shareable private disk (SPD) function permits controlled access by up to 4 different processors to one disk device on which a private disk is mounted. Access is accomplished via direct hardware paths.

The Data Management System of the MRS coordinates access to the shareable disk devices and synchronizes access to the data on the volumes mounted on these disks. Every file on a shareable disk may be accessed at any time by 1 system in write mode or by up to 4 systems in read mode. From a single system, files can also be processed in shared update mode. User programs need not be specially prepared for this function. The fact that files that are to be accessed reside on shareable files is hidden to the user. No special measures need be taken by programs running in different processors and requiring simultaneous access to the same file. Files stored on shareable disks can be cataloged in one or more or all catalogs of the accessing processor. A disk device is considered shareable if it

- is connected to more than one processor (i.e. there are at least 2 access paths via which the disk device may be accessed), and
- was defined as shareable at system generation time.

The concept "shareable", as defined here, is a feature of the disk device and should not be mixed up with the "shareable" feature associated with files or volumes. This disk device feature is entirely defined by the existing hardware configuration and the system generation.

A shareable disk device loses its shareability only if it was defined as "unshareable" during a new system generation and/or reconfiguration of the hardware. Dynamic modification of this feature is not possible.

A volume becomes shareable when it is mounted in a shareable disk device. The shareability feature can be used only in conjunction with private volumes.

Notes on SPD operation

- It is possible for 1, 2, 3 or 4 processors to access one shareable disk, and each of these disks may in turn access other shareable disks.

  Disks mounted on shareable disk devices, as well as files on these disks, should only be those which must actually be accessed simultaneously from several processors.

- A public disk can be mounted in a shareable device but it can be referenced only by its owner (operating system). If it is referenced by more than one processor, the results are unpredictable.

- If a user without remote catalog access (RCA) wants to access a file that was generated by a different processor, he must define this file to his own system (TSOSCAT) beforehand by means of a FILE command or macro. This is not necessary when the RCA function is used.

Shareable private disk

- To minimize the runtime of jobs working with SPD, the following considerations should be borne in mind:

  - Primary and secondary allocation in the FILE command should meet the expected file size requirements.

  - Catalog entries for SPD files which are continually accessed by more than one processor should be kept in all catalogs of the accessing systems. This avoids additional time requirements for communication (data transmission).

- The operand VTOC=YES/NO in the FSTATUS command/macro permits output of the VTOC catalog entries of private disks to be initiated rather than output of the TSOSCAT entries (see the FSTATUS command).

A.6 MEANS OF SUPPORTING DEVICES NOT LOGICALLY SUPPORTED BY BS2000

TOS (tape operating system) runtime parameters

The following TOS runtime parameters are supported by the BS2000 command language. The commands are applicable to files processed by TOS FCP (File Control Processor) or physical I/O (EXCP).

Note:

Commands must contain a slash in column 1 and cannot contain a slash in column 2. (In TOS, both columns 1 and 2 have to contain slashes.) Continuation cards must contain a slash in column 1, and the data must begin in column 2 (not column 16 as in TOS). Otherwise, the command format under an operating system with virtual addressing is identical to that required by TOS.

TOS runtime parameters	Function
CHANGE	Changes the file link name in a Task File Table entry.
FILE	Replaces TOS ASSGN and allows the TOS user to assign private devices for his use.
HOLD	Places an entry in the Task File Table in the HOLD status.

CHANGE

The CHANGE command changes the file link name or the symbolic device name in a Task File Table entry. All other values in this TFT entry remain unaltered.

Operation	Operands
CHANGE	[link1],{link2 | symdev}

link1 — This operand specifies the file link name (1 to 8 bytes).

If this operand is omitted, the first TFT entry with the link name C'␣␣␣␣␣␣␣␣' is processed.

link2 — This operand specifies a new file link name which is to replace the previous link name.

symdev — This operand specifies a new symbolic TOS device name. It is provided for reasons of compatibility and is defined with a device-oriented, non-standard BS2000 FILE command.

FILE (ASSGN)

The TOS FILE command replaces the TOS runtime parameter ASSGN. The TOS format of the FILE command is defined below and conforms to TOS command syntax.

Operation	Operands
FILE	SYMDEV=sdn[,LINK=symbol] [,DEVICE={D5881 | D3455 | D5882 | D3465 | D3468 | D3470 | D3475 | D3480 | D348E | TA | TAPE | T9G | T9N | T9P | WORK}][,MOUNT=mnemonic]

SYMDEV= This operand specifies that this is a TOS FILE command.

This operand contains the symbolic device name (sdn) required by TOS; maximum length: 6 characters.

Note:
This macro will no longer be supported after BS2000 V8.0.

LINK= This operand specifies the file link name.
Although TOS programs do not make use of this operand, it can still be used with other commands (e.g. HOLD, RELEASE).
If the operand is omitted, a link name consisting of blanks (X'40') is generated.
If a FILE command has the same link name as a previous FILE command, the new command replaces the old definition.

Processing takes place in the following sequence:

```
FILE command 1      FILE␣LINK=X, SYMDEV=ABC
                      .
                      .
                      .
                    RELEASE␣X,KEEP
                      .
                      .
FILE command 2      FILE␣LINK=X, SYMDEV=XYZ
```

Any private devices associated with FILE command 1 are returned to the job, not to the system. This permits reuse of these private devices within the job. The devices associated with FILE command 1 are not associated in any way with FILE command 2; they are merely returned to the job.

DEVICE= This operand specifies the device type.

=D3455
=D5881 3455 Disk Storage Unit

=D3465
=D5882 3465 Disk Storage Unit

=D3468 3468 Disk Storage Unit

=D3470 3470 Fixed-Disk Storage Unit

=D3475 3475 Fixed-Disk Storage Unit

=D3480 3480 Fixed-Disk Storage Unit

=D348E 348E Fixed-Disk Storage Unit

=TA 9-track tape with data conversion

=TAPE 9-track tape with maximum recording density

=T9G 9-track GCR (group coded recording)

=T9N 9-track NRZ (no return to zero)

=T9P 9-track PE (phase encoded)

=WORK This operand specifies the system work tape. The system requests 9-track system work tapes, previously specified by the operator (via the SETUP command).

If no tape is available, this operand is equivalent to the TAPE operand.

Default value:
If no operand is specified, TAPE is assumed.

MOUNT= This operand specifies the installation name of a particular device as a 2-character mnemonic. This allows the user to specify a preference for a specified device. If this operand is specified, the DEVICE operand is ignored.

Default value:
The system selects an appropriate device according to the device type specified in the DEVICE operand.

Note:

This macro will no longer be supported after BS2000 Version 8.0.

HOLD

The HOLD command places an entry in the Task File Table (TFT) in the HOLD status. This serves to defer a subsequent RELEASE command (or REL macro) for this TFT entry until a DROP command with the relevant file link name is issued.

Operation	Operands
HOLD	[{link / link,TOS / TOS}]

link — This operand specifies the file link name of a TFT entry to be placed in the HOLD status. If no entry with this name existed previously, a new TFT entry is created for this link name. Further entries can be made in this TFT entry by means of a subsequent FILE command. If "link" is not specified, the first TFT entry with the link name C'␣␣␣␣␣␣␣␣' is processed.

TOS — This operand specifies that the TFT entry in question is for a TOS file, a file type which exists for reasons of compatibility. It is created with a non-standard FILE command.

TOS macros in the DMS

The following macros, related to file processing and device management, are available to DMS programs. They may be used for class I programs only. FCB macros valid for TOS may also be used for class I programs, but they are not discussed here.

TOS macro	Function
ASSGN	allows dynamic device assignment for TOS programs.
CCB	allows physical level I/O specifications for a CCB.
CCBD	generates a DSECT for the CCB.
CHNGE	changes the file link name.
DDEV	allows a program to release a specified device.
EXCP	initiates (or queues) a user's I/O channel program.
EXCPW	same as EXCP except that the user is pended until his channel program has been completed.
QUIET	places the calling program in a wait state until its I/O activity has been completed.
WAIT	places the job in a wait state until the I/O operation defined by the CCB has been completed.

A

Dynamic device assignment (type D)

Operation	Operands
ASSGN	ccbaddr

ccbaddr — This operand specifies the address of the CCB containing the symbolic device name of the device to be assigned (relexp).

Note:

This macro will no longer be supported after BS2000 Version 8.0.

CCB Command Control Block (type 0)

Operation	Operands
CCB	symdev,ccw,flags,device-type

symdev — This operand specifies the symbolic name of the device with which the CCB is associated (symbol).

ccw — This operand specifies the address of the first CCW to be referenced in this CCB (relexp).

Note:

If data chaining is employed, the restrictions laid down in the relevant peripheral control descriptions must be observed.

flags — This operand specifies the user flags (absexp).

Significance of the flags:

X'20' Skip device error handling
X'10' Error is accepted
X'04' Suppress device error handling messages
X'02' Card punch handling/SDLDR already defined

device-type — This operand specifies the device type (absexp).

Note:

This macro will no longer be supported after BS2000 Version 8.0.

CCBD DSECT for CCB

Operation	Operands
CCBD	D [,{n | *}]

D — This operand is mandatory and specifies that a DSECT is to be generated.

n — This operand is optional; any alphabetic character can be specified for "n". It serves to specify a freely selectable character prefix to be linked to all names in the DSECT.

Default value: I

* — This entry specifies that no prefix is to be used.

Note:

This macro will no longer be supported after BS2000 Version 8.0.

CHNGE - Modify TFT entry (type S)

The CHNGE macro changes the file link name or symbolic device name of a File Control Block. All other TFT entries remain unaltered.

Operation	Operands
CHNGE	[link1],{link2 | symdev}

For a description of the operands, see the CHANGE command.

DDEV - Deallocate device (type O)

Operation	Operands
DDEV	sdn

sdn — This operand specifies the 6-byte symbolic device name of the device to be released.

Note:

This macro will no longer be supported after BS2000 Version 8.0.

If a TOS FILE has placed the device (symbolic device name) in the HOLD status, the device is not deallocated. It is deallocated only after a DROP command has been issued.

EXCP - Execute channel program (type O)

The EXCP macro performs the same functions as EXCPW, except that control is returned to the user after the channel program has been initiated or queued.

Operation	Operands
EXCP	ccb

ccb — This operand specifies the name of the CCB which indicates the I/O operation to be initiated (relexp).

Note:

This macro will no longer be supported after BS2000 Version 8.0.

EXCPW - Execute channel program and wait (type 0)

The function of the EXCPW macro is to initiate (or queue) the user's I/O channel program as specified by the CCW address in the CCB. The user is then placed in the wait state until the channel program has been completed.

Operation	Operands
EXCPW	ccb

ccb This operand specifies the name of the CCB which indicates the I/O operation to be initiated (relexp).

Note:

This macro will no longer be supported after BS2000 Version 8.0.

QUIET - Wait for completion of all I/O operations (type 0)

This macro suspends the calling program until all input/output activities have been terminated.

Operation	Operands
QUIET	

Note:

This macro will no longer be supported after BS2000 Version 8.0.

WAIT - Wait (type 0)

This macro is used to place the task in the wait state until all I/O activities have been completed.

Operation	Operands
WAIT	ccb

ccb This operand specifies the name of the CCB associated with the device on which completion of I/O activities is awaited (relexp).

Note:

This macro will no longer be supported after BS2000 Version 8.0.

A.7 DEVICE TYPE CODES

Family code	Device type code	TOS code	Devices
00			**Operator consoles**
	01	1A	Operator consoles in Systems 7.500, 7.700: 3020, 3021, 3023 Central Operator Console 3020-10, 3021-10, 3023-10 Subconsole 3030, 3033-01 Console Printer (optional hardcopy device)
	02	--	Operator consoles in Systems 7.500, 7.700: 3024, 3026 Central Operator Console 3024-10, 3025-10, 3026-10, 3026-20 Subconsole
		--	3027-11, 3027-21 Operator Consoles
10			**Card readers**
	12	06	237 Card Reader
	14	09	4235, 4239-10, 3150-01, 4239-20, 3150-02, 4239-30, 3150-03 Card Reader
	1A	--	F617D Card Reader
20			**Printers**
	21	01	4245, 4247, 243-10, 243-20, 243-30, 243-31, 4241-52, 3340-01, 4241-54, 3340-02 Printer (132 characters)
	22	02	4242-92, 3340-03, 4242-94, 3340-04, 243-40, 243-41 Printer (160 characters)
	23	12	3341, 3342, 3343 Printer
	24	18	3352 Laser Printer
	25	--	3333, 3336 Printer
	26	--	3337-51, -52, 71, 3338-51, 52 Printer
30			**Card punches**
	31	04	234 Card Punch
	32	05	236 Card Punch
	34	04	4238, 3160 Card Punch
40			**Paper tape devices**
	41	07	Paper tape reader
	42	07	Paper tape punch
	4F	--	Document reader
50			**Miscellaneous devices**
	52	1F	3065 Switch
	53	1B	TRANSDATA 8170 Cluster Controller (Local)
	54	1B	Plotter
60			**Teleprocessing**
	61	0D	968x Front-End Processor
	62	--	968x Front-End Processor
	63	--	968x Front-End Processor
	6E	16	627 Data Exchange Controller
70	71-7F	--	**Physically supported devices**
		--	Plotter, document reader, graphic terminal, etc.

A

Device type codes

Family code	Device type code	TOS	Devices
90			**Floppy disk I/O units**
	91	--	3170 Floppy Disk I/O Unit
	92	--	30243, 30244, 30230, 30263 Floppy Disk I/O Unit
	92	--	3171 Floppy Disk I/O Unit
A0			**Disk storage units**
	A6	6C	3470 Fixed-Disk Storage Unit (not with 4004/151), 3472 Fixed-Disk Storage Unit
	A8	5C	3454, 3455 Disk Storage Unit
	A9	7C	3464, 3465 Disk Storage Unit
	AA	--	3468 Disk Storage Unit
	AB	--	3475 Fixed-Disk Storage Unit
	AC	--	3480 Fixed-Disk Storage Unit
	AD	--	348E Fixed-Disk Storage Unit
B0			**Magnetic tape devices, 9-track, unimodal**
	B1	0B	432, 442, 4443, 4446 Magnetic Tape Device, NRZ, 800 bpi
	B2	0A	451, 453, 4420, 3570, 3571, 3530, 3531 Magnetic Tape Device, PE, 1600 bpi
	B4	0E	3557, 3559 Magnetic Tape Device, GCR, 6250 bpi
E0			**Magnetic tape devices, 9-track, bimodal**
	E1	--	450, 3550, 454, 3554, 4453, 3540, 3521, 3523 Magnetic Tape Device, NRZ/PE, 800/1600 bpi
	E2	0E	3557, 3559, 3525, 3526, 3513, 3516 Magnetic Tape Device, PE/GCR, 1600/6250 bpi
	E3	--	3518, 3528 Streaming Tapes

LITERATURE

BS2000
Control System Command Language
Reference Manual

BS2000
DMS Tape Processing
Reference Manual

BS2000
Executive Macros
Reference Manual

BS2000
MSCF Multiprocessor System
Reference Manual

BS2000
RFA
Reference Manual

BS2000
System Administrator's Guide
Reference Manual

BS2000
Systemanwendung **
Reference Guide

BS2000
System Exits
Reference Manual

BS2000
System Messages
Reference Manual

BS2000
System Operator's Guide
Reference Manual

BS1000/BS2000
Systems Standards
Reference Manual

Druckschriftenverzeichnis *
(List of Siemens publications)

* available under German title only
** currently available in German only

U805-J-Z55-5-7600

GLOSSARY

The following terms are used throughout this publication and are defined here for the convenience of the reader. The majority of these terms are also defined where they first appear in the text.

access method
A conventional data management technique that defines to the user the organization of data and the method of data transfer between an input/output device and the main memory of the system. Examples of DMS access methods are ISAM (Indexed Sequential Access Method) or UPAM (User Primary Access Method.

alphanumeric
Letters A through Z and numerals 0 through 9.

batch job, non-interactive job
A job started with the aid of ENTER.

block
The unit of transfer to or from an I/O device. For example, the data between two interblock gaps on magnetic tape is a block.

buffer
A contiguous area of memory. A portion of main memory from which data is read or into which it is written.

catid
Catalog ID of a pubset; cf. PVSID.

class I program
Programs that are restricted to the physical main memory addressing capacity of the system. These programs must reside in main memory throughout their execution, and require that main memory be assigned to them contiguously.

class II program
Programs that do not require contiguous main memory for execution and reside in the system's virtual memory. These programs are broken up into pages (4096 bytes), use the virtual addressing facility of the system, and are normally paged between the paging memory and main memory during their execution.

class 1 memory
That portion of virtual memory that is occupied by those modules of the Executive which are resident in main memory. All class 1 pages are marked privileged and non-pageable. No paging memory image exists and these pages are resident in main memory throughout the session.

class 5 memory
That part of the user's virtual memory containing the pageable areas required for a user job that are allocated dynamically by the Executive.

class 6 memory
That part of the user's virtual memory containing the user programs that are allocated dynamically by the Executive.

Command Control Block (CCB)
An area in a user program in which data is passed between the user and the input/output device accessed by the program. The CCB is used for physical level I/O programming.

CSECT (control section)
A part of a program that can be relocated at load time, independently of other parts of the program and without impairing or changing the program logic. CSECTs are the basic input unit to the Linkage Editor program.

DSECT (dummy section)
A named control section that is assembled but is not part of the object program. It is a convenient means of describing the layout of a storage area without actually reserving storage space.

DTF (define the file)
A tape operating system (TOS) mnemonic for the general class of TOS file definition macros. For example, DTFSR stands for "define the file serial" and is a TOS file definition macro for sequentially organized files. BTAM and SAM constitute sequential access methods in an operating system with virtual addressing.

ETX (end of text)
A character signifying the end of a data transmission. The actual bit configuration of this character varies with the type of terminal being used. ETX is also known as "end marker" (EM).

EXCP (execute channel program)
A macro on the physical I/O level that allows the programmer to communicate with an I/O device without using the standard DMS or TOS FCP logic.

FCB

Two meanings:

1. File Control Block - contains all information relating to a given file;

2. Macro for generating the File Control Block.

FIFO (first in first out)
A queuing technique that processes information in the same sequence as it entered the queue.

File
Related records combined to form a named entity. These identifiable entities are called "files". For example, the following are files:

1. Conventional input/output program data.

2. Load modules and object module libraries.

3. Textual information generated and processed by the File Editor.

foreign volume
In the DMS, a foreign volume is one that is not cataloged in the system.

home address, hardware key field
Part of the track descriptor record (record 0) or data record found in each track of a disk. The key field contains control information concerning the record. It can vary in size from 1 to 255 bytes.

interactive job
An interactive job (also known as a conversational job) is one that is initiated and run from a terminal in dialog with the user.

I/O, Logical Level
A type of input/output processing where the user operates on record level by means of the GET and PUT macros, or of other macros. Device handling is effected using the OPEN and CLOSE logic. The blocking/deblocking of records as well as error recovery are also carried out by the system routines.

I/O, physical Level
A type of input/output processing where the user operates on device level. He supplies the Command Control Blocks (CCBs) and Channel Command Words (CCWs) and controls the devices himself in conjunction with the EXCP macro (Execute Channel Program). The blocking/deblocking of records as well as error recovery are also the user's responsibility.

job
The sum total of everything that occurs between a user's LOGON and LOGOFF commands. Whether the job is already completely defined (batch mode) or whether the individual steps are not defined until the job is processed (interactive mode) is of no consequence here.

Link name
This is the logical file name that provides the connection between the FCB (File Control Block) macro and the job control language (JCL), and between the Task File Table (TFT) and device handling.

Locate mode
In locate mode means the user requests the location of the current record in the buffer area. The user is responsible for transferring data to and from the buffer.

move mode
In move mode means the user specifies the location of the record in his program and the system is responsible for moving it to or from the buffer.

null file
A file that is logically empty. This file has been cataloged by the system and is assigned storage space by the system, but it contains no data.

PAM block, PAM page
Contiguous memory of 2048 bytes, starting at an address divisible by 2048.

privileged mode, program
All operating system components which do not execute in processor state P1. The four processor states are:

1. **P1 (processor state)**: interprets and executes the user programs, is therefore also often referred to as user state.

2. **P2 (interrupt response state)**: performs certain functions depending on the interrupt status.

3. **P3 (interrupt control state)**: analyzes the cause of an interrupt, establishes its priority, and transfers control to the related interrupt handling routines.

4. **P4 (machine condition state**: is entered whenever a machine error occurs.

procedure file
A set of predefined operations/commands that specify a sequence to be performed. A procedure file may also contain data to be used as input for a program. The two types of procedure file are the ENTER and the DO file. Both can be initiated from a terminal or from a batch job. However, when an ENTER file is initiated, the calling task does not remain connected to the procedure, as it does when a DO file is initiated. The ENTER file becomes a background (non-interactive) task.

PVS - Public Volume Set or pubset
A set consisting of a number of independent public volumes (pubsets) on a single system.

PVS ID
Identification of a pubset. The fourth character of the volume serial number (vsn) is used for identification purposes. In the file name, this is indicated by means of ":x:".

runtime parameter
A term adopted from TOS (tape operating system) and used for parameters needed during TOS program execution. These parameters can be used to allocate devices to TOS programs, to position files, to check standard labels etc.

shareable file
A file that the user catalogs with the SHARE=YES operand. Files marked as shareable can be accessed by any user who can provide the user ID of the file's creator, the file name, and (if required) the appropriate password by which the file is protected.

spoolin
The transcription of input from low-speed peripherals to temporary disk files.

spoolout
The transcription by the software product SPOOL of output from the system to relatively low-speed devices such as printers, card punches etc.

step
A logical subdivision of a job. Abnormal termination of a step in a job causes the next step to be executed.

symbolic device name
A TOS name associated with each logical file in a TOS program and used to assign the actual I/O device to this file. As a rule, the name is six characters in length , e.g. SYS007. This is not the same as the installation mnemonic for a device, which is a two-character code such as A1, assigned at system generation time.

system files (SYSOUT, SYSLST, SYSLSTnn, SYSOPT, SYSCMD, SYSDTA, SYSIPT)
Users have only indirect access to system files; the Control System routines can access them directly. System files provide data and facilities required by the Control System functions. The two general classes of system files are referred to as temporary and symbolic. Both classes are assigned standard names.

Temporary system files are created and managed by routines that use the facilities of the job control component of BS2000. Their logical names are:

SYSOUT — system messages that record the commands and their responses and are destined for output to printer or terminal.

SYSLST — listings such as those produced by an assembly or Linkage Editor run, which are destined for output to printer.

SYSLSTnn — listings; up to 99 system files can be specified. Each listing must be assigned to a cataloged file.

SYSOPT — files in card format, such as modules produced by an assembly run, which are destined for output to punch.

These temporary system files are given temporary file names when they are first accessed by a user or when they are accessed by the Control System via OPEN. After spoolout these files are deleted and their disk space is restored to the system.

Other system files managed by the system are strictly symbolic in nature. They never receive file names, nor do they reside on disk. SYSCMD, SYSDTA and SYSIPT are such symbolic files.

SYSCMD — comprises the commands issued by a user in his job. The Control System alone reads this file.

SYSDTA — comprises the input data for a module and is accessed by the RDATA macro.

SYSIPT — is the symbolic system input file maintained for reasons of compatibility with TOS. It is the TOS counterpart of the SYSDTA file and is accessed by the TOS macro RDCRD.

TM (tape mark)
A single character on magnetic tape, preceded and followed by a gap, that is used to delimit data groups. In the case of 9-track tape, the bit configuration depends on the system.

volume serial number (vsn)
A six-character number assigned to a volume at initialization time (VOLIN or INIT) and included in the standard volume label for identification purposes.

volume table of contents (VTOC)
An extent which resides on a disk and contains file labels and file address information.

From

Name _____

Company _____

Street _____

City/Postal Code _____

Phone _____

Date _____

To
Siemens AG

K D ST QM 2
Manualredaktion

Otto-Hahn-Ring 6
D-8000 München 83

Reader's Reply Form

BS2000
DMS Disk Processing
Reference Manual
Edition November 1985 (BS2000 V8.0A)
Order No. U805-J-Z55-5-7600

Page	Criticisms/Suggestions/Corrections

Page	Criticisms/Suggestions/Corrections

From

Name _____

Company _____

Street _____

City/Postal Code _____

Phone _____

Date _____

To
Siemens AG

K D ST QM 2
Manualredaktion

Otto-Hahn-Ring 6
D-8000 München 83

Reader's Reply Form

BS2000
DMS Disk Processing
Reference Manual
Edition November 1985 (BS2000 V8.0A)
Order No. U805–J–Z55–5–7600

Page	Criticisms/Suggestions/Corrections

Page	Criticisms/Suggestions/Corrections

Published by Bereich Datentechnik
Postfach 830951, D-8000 München 83

Siemens Aktiengesellschaft

BS2000

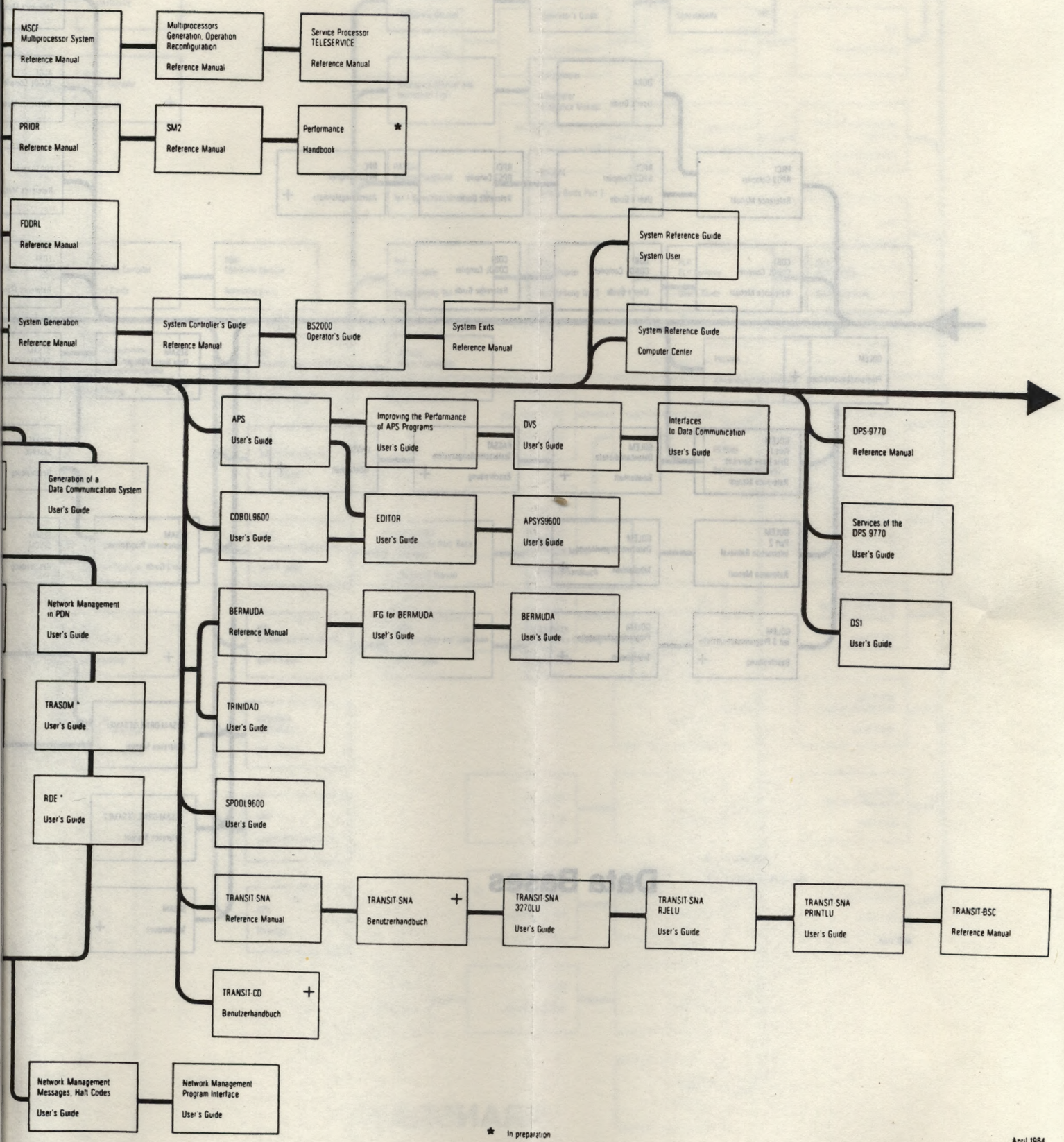
- System Messages — Reference Manual
- Systems Standard — Reference Manual
- DMS Tape Processing — Reference Manual
- DMS Disk Processing — Reference Manual
- Executive Macros — Reference Manual
- LEASY — Reference Manual
- RFA — Reference Manual
- LMS — Reference Manual
- Laser Printer — Reference Manual
- ARCHIVE — Reference Manual
- Utility Routines — Reference Manual
- EDT — Reference Manual
- EDOR — Reference Manual
- SORT — Reference Manual
- PERCON — Reference Manual
- Control System Command Language — Reference Manual
- Job Variables — Reference Manual
- Interactive Debugging Aid — Reference Manual
- EASP + — Beschreibung
- Introductory Guide for System Users — User's Guide
- Introductory Guide for System Operators — User's Guide

TRANSDATA

- Systems Standards — Reference Manual
- Directions for the Terminal User — User's Guide
- 9749, 9750, 9752 Data Display Terminals Programmer's Interface — User's Guide
- 9001 Printer Programmer's Interface — User's Guide
- 9003 Printer Programmer's Interface — User's Guide
- 9004 Printer Programmer's Interface — User's Guide
- DCAM Program Interfaces — Reference Manual
  - DCAM COBOL Calls — User's Guide
  - DCAM Macros — User's Guide
- Connection to Packet Switching Networks (X.25, X 29) — Reference Manual
- UTM Program Interfaces — Reference Manual
  - UTM KDCS COBOL Calls — User's Guide
    - UTM Generation and Administration — User's Guide
      - UTM-RPG1 UTM for RPG Users — User's Guide
- TIAM — User's Guide
- FHS — User's Guide
  - IFG for FHS — User's Guide
- ASMUS — User's Guide
  - ASMUS2 + — Benutzerhandbuch
- RCOM — User's Guide
  - RCOM-S — Reference Manual
    - RCOM FTP — Reference Manual
- File Transfer — User's Guide
- Generation of Preferred Configurations — User's Guide
- Network Management in BS2000 — User's Guide
- NEOSP — User's Guide
- TDA * — User's Guide
- ADS + — Benutzerhandbuch
- RBAS — User's Guide
- OMNIS — User's Guide

Publications Plan for

- BS2000
- TRANSDATA
- Programming Systems
- Data Bases

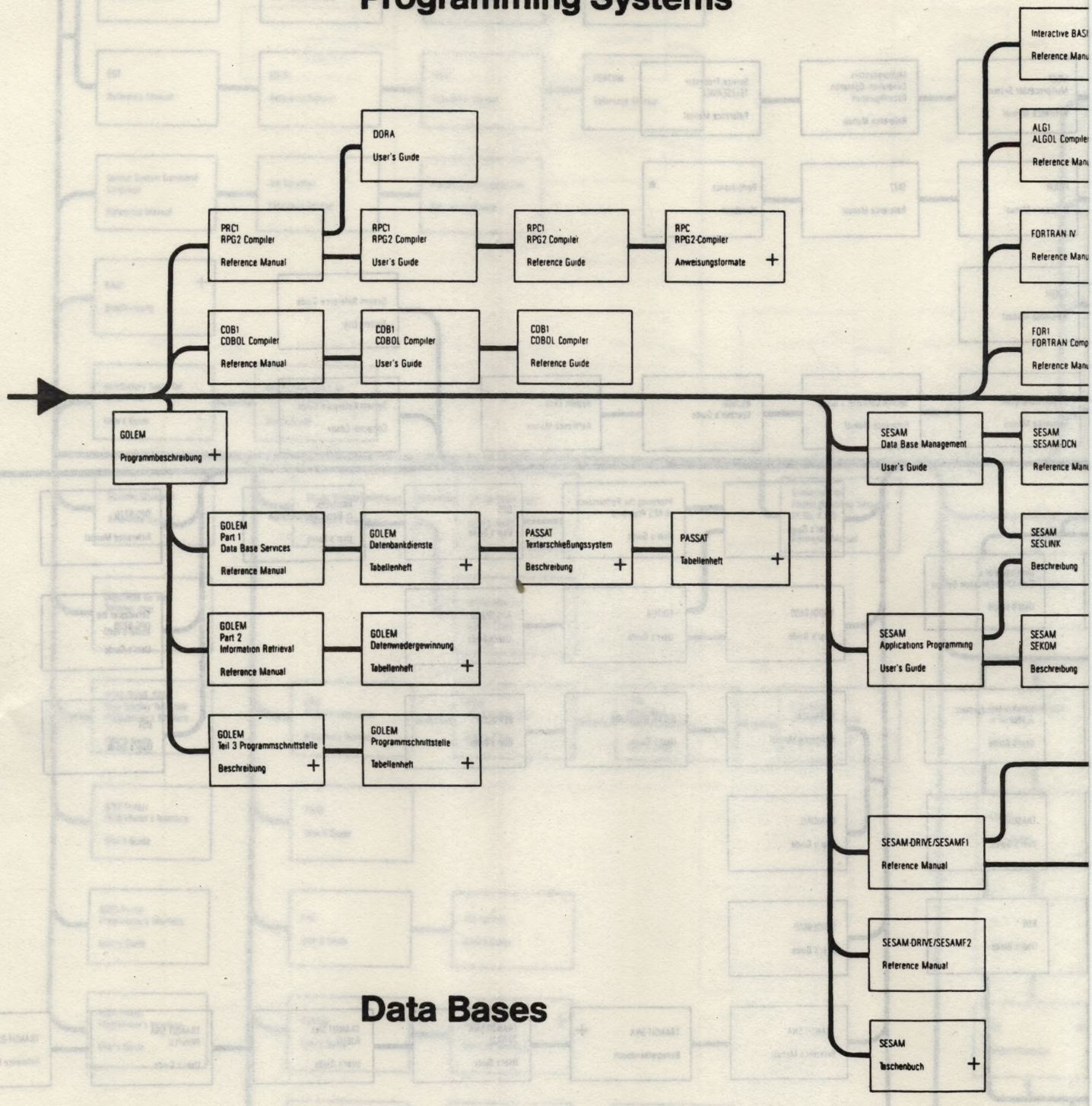


* In preparation

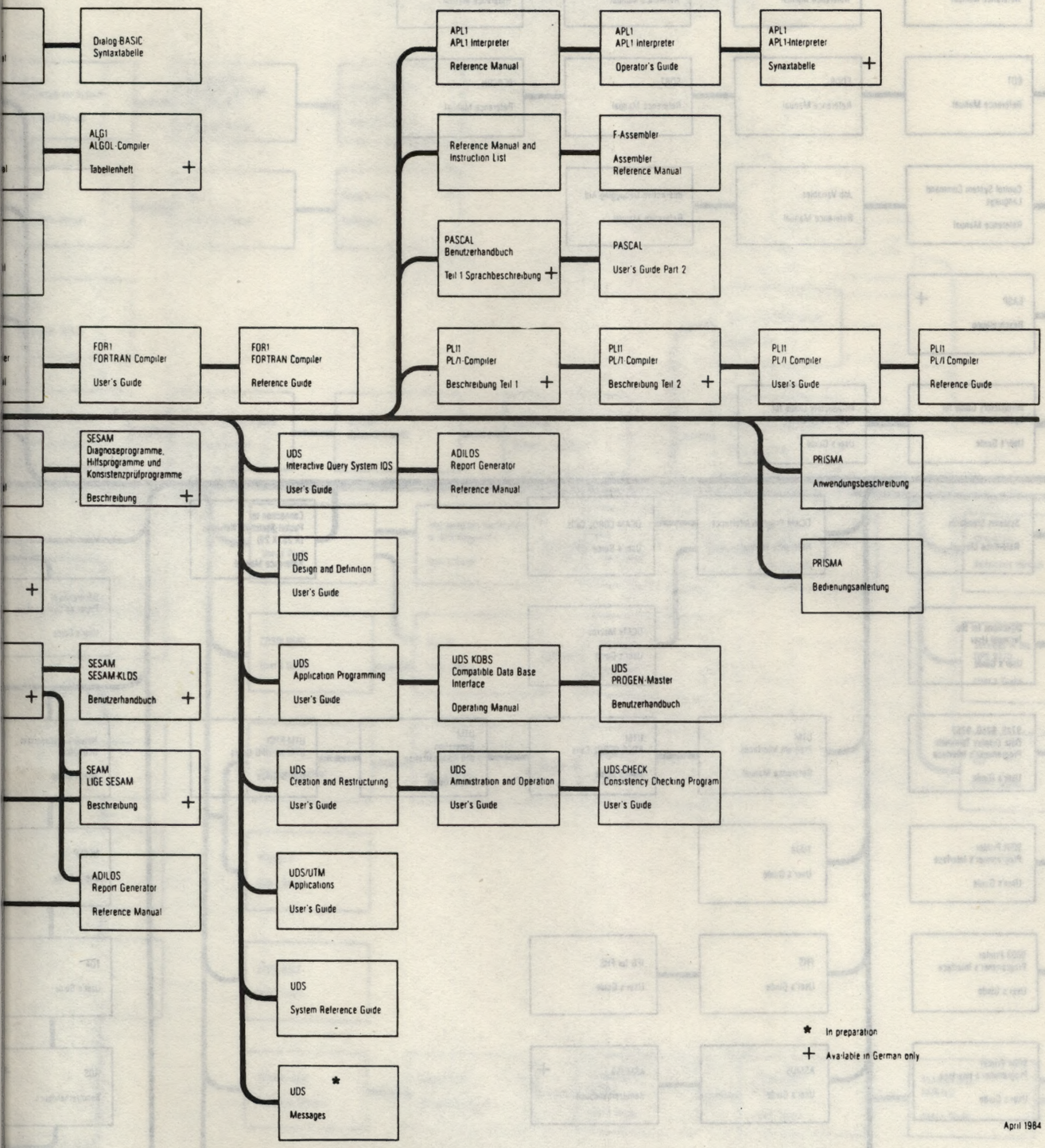
\+ Available in German only

April 1984

Programming Systems



Data Bases



* In preparation

\+ Available in German only

April 1984